2005-06

# IPSec-based dynamic security services for the MYSEA environment

Horn, John F.

Monterey, California. Naval Postgraduate School


# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

# THESIS

**IPSEC-BASED DYNAMIC SECURITY SERVICES FOR THE MYSEA ENVIRONMENT**

by

John F. Horn

June 2005

| | |
|---|---|
| Thesis Advisor: | Cynthia E. Irvine |
| Co-Advisor: | Thuy D. Nguyen |

**Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

| REPORT DOCUMENTATION PAGE | | | Form Approved OMB No. 0704-0188 |
|---|---|---|---|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503. | | | |
| **1. AGENCY USE ONLY** ***(Leave blank)*** | **2. REPORT DATE** June 2005 | **3. REPORT TYPE AND DATES COVERED** Master's Thesis | |
| **4. TITLE AND SUBTITLE**: IPsec-Based Dynamic Security Services for the MYSEA Environment | | | **5. FUNDING NUMBERS** |
| **6. AUTHOR(S) John F. Horn** | | | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000 | | | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)** N/A | | | **10. SPONSORING/MONITORING AGENCY REPORT NUMBER** |
| **11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. | | | |
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release; distribution is unlimited | | | **12b. DISTRIBUTION CODE** |


**13. ABSTRACT (maximum 200 words)**

It is recognized that security services in information-processing systems require access to finite resources in the execution of their duties. In response to the changing threats faced by a system and/or the availability of system resources, it is desired that the system be able to adjust its operational security policies automatically while continuing to function under an acceptable global security policy.

This work involves the analysis and integration of a dynamic security service (DSS)-enabled IPsec implementation into a form ready for installation into the MYSEA environment. The feasibility of dynamic security services is demonstrated with support for secrecy and/or integrity protection of MLS server-to-end-user communication via a Trusted Path Extension. This is accomplished through the modulation of the IPsec security associations to adapt to operational needs.

The result of this research is beneficial to Homeland Security, the Department of Defense, and the intelligence community by enabling remote distributed computing clients to operate in a secure manner that remains flexible to adapt to changing requirements of protection on the network and the availability of resources on terminating hosts. Furthermore, these methods can aid the realization of high-assurance edge-client connectivity in the creation and extension of the Global Information Grid (GIG).


| **14. SUBJECT TERMS** Information Assurance, Multilevel Security, Dynamic Security, Monterey Security Architecture, IPsec, KeyNote | | | **15. NUMBER OF PAGES** 132 |
|---|---|---|---|
| | | | **16. PRICE CODE** |
| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** UL |

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited**

# IPSEC-BASED DYNAMIC SECURITY SERVICES FOR THE MYSEA ENVIRONMENT

John F. Horn
Civilian, Naval Postgraduate School
B.S., University of Akron, 1999

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

**NAVAL POSTGRADUATE SCHOOL**
**June 2005**

Author: John F. Horn

Approved by: Cynthia E. Irvine, Ph.D.
Thesis Advisor

Thuy D. Nguyen
Co-Advisor

Peter J. Denning, Ph.D.
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

It is recognized that security services in information-processing systems require access to finite resources in the execution of their duties. In response to the changing threats faced by a system and/or the availability of system resources, it is desired that the system be able to adjust its operational security policies automatically while continuing to function under an acceptable global security policy.

This work involves the analysis and integration of a dynamic security service (DSS)-enabled IPsec implementation into a form ready for installation into the MYSEA environment. The feasibility of dynamic security services is demonstrated with support for secrecy and/or integrity protection of MLS server-to-end-user communication via a Trusted Path Extension. This is accomplished through the modulation of the IPsec security associations to adapt to operational needs.

The result of this research is beneficial to Homeland Security, the Department of Defense, and the intelligence community by enabling remote distributed computing clients to operate in a secure manner that remains flexible to adapt to changing requirements of protection on the network and the availability of resources on terminating hosts. Furthermore, these methods can aid the realization of high-assurance edge-client connectivity in the creation and extension of the Global Information Grid (GIG).

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

# LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

# ACRONYMNS AND ABBREVIATIONS

| | |
|---|---|
| CC | Common Criteria |
| COTS | Commercial Off-The-Shelf |
| DAC | Discretionary Access Control |
| DSS | Dynamic Security Service |
| EAL | Evaluation Assurance Level |
| HTTP | HyperText Transfer Protocol |
| IKE | Internet Key Exchange |
| ISAKMP | Internet Security Association and Key Management Protocol |
| IP | Internet Protocol |
| LAN | Local Area Network |
| MAC | Mandatory Access Controls |
| MLS | Multilevel Secure |
| MYSEA | Monterey Security Architecture |
| NAT | Network Address Translation |
| QoS | Quality of Service |
| QoSS | Quality of Security Service |
| SIPRNET | Secret Internet Protocol Router Network |
| STOP | Secure Trusted Operating Program |
| TCM | Trusted Channel Module |
| TCP | Transmission Control Protocol |
| TPE | Trusted Path Extension |
| UDP | User Datagram Protocol |

THIS PAGE INTENTIONALLY LEFT BLANK

# ACKNOWLEDGMENTS

I thank my thesis advisors Thuy Nguyen and Cynthia Irvine for dedicating time, patience, resources, and expertise to support of this project. I give special thanks to my wife who has graciously supported my graduate studies at the Naval Postgraduate School.

This material is based upon work supported by the National Science Foundation under Grant No. DUE-0114018. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

## A. MOTIVATION

Designers and maintainers of computing systems responsible for storing and processing critical and/or sensitive information are faced with the challenges of controlling access to and modification of data contained within the computing system as well as data in transit between systems. Threats to the security of the system often change as capabilities of adversaries improve and operating environments evolve. Protection mechanisms, such as the deployment of IPsec for securing network communications, must often be reconfigured in response to these changing threats.

Likewise, it is recognized that security services in any information-processing system require access to finite resources in the execution of their duties. Since a shortage of resources such as CPU, memory, and/or time has the potential to adversely affect the execution of critical processes in the system, management of available resources can be critical. The dynamic nature of both resource availability and the threat level faced by an information-processing system has resulted in the need for the operational security policies to be configured with multiple security profiles that can be selected based on the state of the current operating environment. The security mechanisms responsible for adapting the system's operational needs to the changing security policies is referred to as Dynamic Security Services.

The motivation for this study is to improve the protection of sensitive communications in the MYSEA multilevel secure environment while providing for dynamic modulation of security resources in a manner consistent with the system operating policy.

## B. PURPOSE

The goal of this project is to produce a dynamic security service-enabled IPsec subsystem for the MYSEA environment. The implementation must incorporate adaptable secrecy and integrity protection for both sensitive authentication and integrity-critical information delivery tasks in the MYSEA environment. The implementation will both provide a functional interface to specify the current operating policy of the system

and support the enforcement of the policy. The system will then be tested to ensure correct functionality of the implementation.

## C. ORGANIZATION OF PAPER

A brief introduction to the paper is presented in Chapter I. Chapter II lays down sufficient background describing both the purpose of this thesis and how it fits with key characteristics of the MYSEA project. Coverage includes IPsec, Dynamic Security Services (DSS), and the role of DSS in TPE-MYSEA server communication. Chapter III lays out the requirements of a dynamic security service-enabled IPsec implementation in the MYSEA environment. These requirements are then used to form a four-stage plan designed to reduce complexity in the implementation of the system. A discussion of implementation difficulties is also included. Chapter IV provides testing coverage which is used to verify correct functionality of the system, and Chapter V reviews the product of this effort and details follow-on work.

Four appendices are also included. Appendix A and B detail the steps required for installation, configuration, and a demonstration of the system. Appendix C contains the configuration files used in the Stage 3 & 4 implementation with explanations. Appendix D contains the test procedures that correspond to the testing described in Chapter IV.

# II. BACKGROUND

This chapter provides background material relating to this project. The first section is an overview of the MYSEA project, the second section contains a brief review of IPsec, and the third introduces the concept of dynamic security services. Finally, there is a discussion on the integration of dynamic security services into the MYSEA environment.

## A. MYSEA ENVIRONMENT OVERVIEW

The Monterey Security Architecture has been designed to serve as a complete, yet flexible, trusted and distributed multilevel secure (MLS) environment [IRV04]. While MLS systems have existed within the DoD and commercial marketplace for many years, the combination of the high cost of acquiring, configuring, and maintaining such componentry, and the difficulty of (re)training end-users in the use of current and historical MLS systems have lead to sluggish adoption of such systems. As an alternative, in environments ranging from corporate R&D to national intelligence and law-enforcement, current practice in many agencies that have strong requirements for information secrecy and integrity is to use low trust systems for all assets with physical separation between "high-value" systems and systems operating at lower secrecy and integrity levels. This is referred to as a "system high" mode of operation. While in the past, such isolated, single-level computers and networks generally proved sufficient for completion of mission, in the age of the Global Information Grid (GIG) [LEO00], a new solution which provides for controlled information sharing across enterprises is required.

MYSEA's design addresses user acceptability by utilizing commercial-off-the-shelf (COTS) components to handle tasks such as user-interface, desktop applications, and network switching and routing while using high-assurance components where needed for total, system assurance. An architectural network view of the MYSEA environment is depicted in Figure 1. Client access is handled via diskless, Intel-based personal computers running operating systems such as Linux or Windows variants. By using a common enterprise operating environment on the desktop, cost and availability issues associated with the purchase and construction of end-user applications suitable for execution in a traditional MLS environment are greatly reduced.

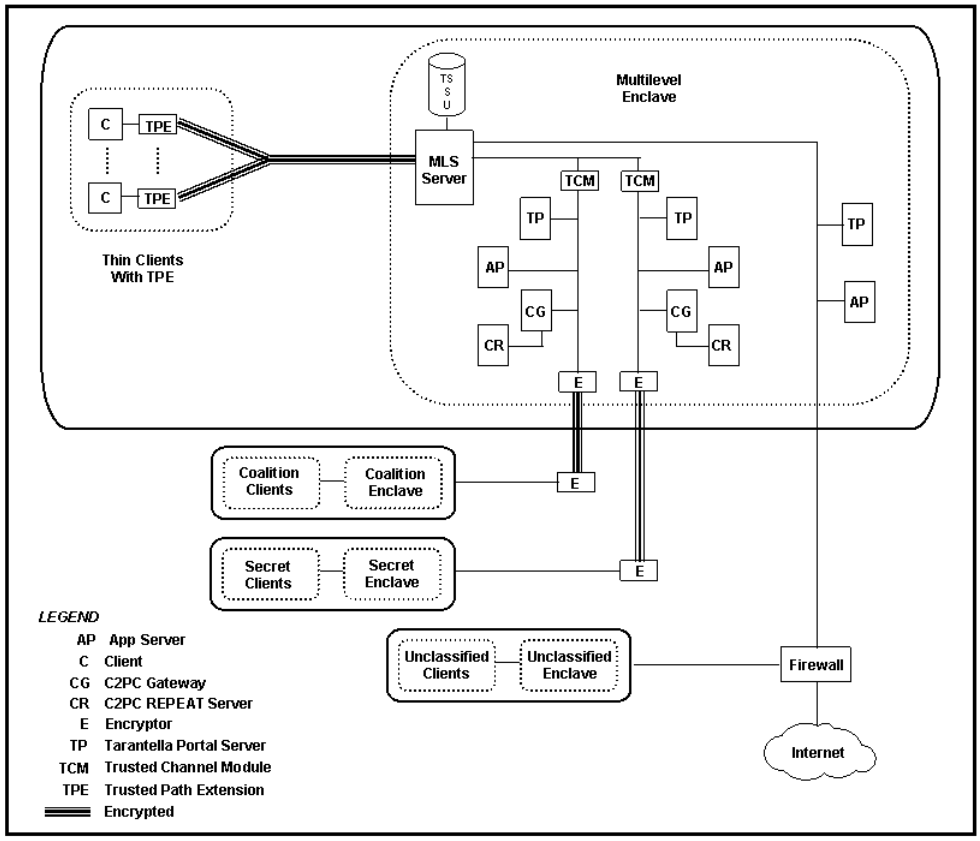


Figure 1. MYSEA Environment Topology

The storage and exchange of information between the unevaluated, untrusted or "less-trusted" components is directed and supervised by a high-assurance, network-connected MLS server, currently the DigitalNet XTS-400 running the high-assurance, general-purpose Secure Trusted Operating System (STOP). The STOP enforces mandatory access control (MAC) as well as traditional Unix-like discretionary access control (DAC). The MAC capabilities of the STOP include an implementation of the Bell-LaPadula policy [BEL76] for secrecy and the Biba policy [BIB77] for integrity enforcement. Both models have been mathematically proven to be complete and secure, and DigitalNet's implementation has completed evaluation at Common Criteria [CC04] EAL 4(+) [NIA04]. Additionally, the XTS-400/STOP combination is currently undergoing further evaluation at EAL 5+ [CYG04]. From this point forward, this component of MYSEA will be referred to as the "MYSEA server" or simply the "MLS server".

Network communication between the MYSEA server and each remote thin client is enabled via an inline Trusted Path Extension (TPE), typically co-located with the client. The TPE creates an encrypted, unforgable communication and control channel (trusted path) between the thin client and the MLS server. In this LAN, protected client-MLS server communication, including authentication and session level negotiation are afforded by the TPE. Unevaluated, and thus potentially malicious, networking equipment may be juxtaposed between the TPE and the MLS server. Other communication on the same LAN between the MLS server and other TPEs can simultaneously take place at a wide variety of session levels without compromising the integrity or confidentiality of the protected channels maintained by a given TPE. The TPE is targeted to run as a specialized hardware appliance executing a custom high-assurance (EAL 7) security kernel currently in development at the Naval Postgraduate School. Currently, a TPE prototype hosted on a Pocket PC running Linux has been implemented.

Finally, it has been recognized that a requirement for communication between the MYSEA server and single-level networks exists. Such networks may be "legacy" networks that connect to the kinds of single-level systems referred to above. Alternatively, these may be ad-hoc or "coalition" networks designed for streamlined information sharing with allies and strategic partners. Currently, direct connectivity to such networks in the MYSEA environment is permitted by assigning an appropriate STOP security label to a specific network interface connected to the single-level network. Subsequent communication on that network interface is then treated as communication with a user or process operating at the secrecy and integrity level assigned to the interface. The MYSEA architects envision adapting and expanding the use of the TPE hardware and kernel to a Trusted Channel Module (TCM) that is able to securely bridge or multiplex multiple single-level network connections to a lesser number of MLS server network ports. Additional information on the design and requirements of the TCM and its associated communication protocol can be found in Sears [SEA04].

Again, it is emphasized that the primary specialized componentry in the MYSEA environment besides the MLS server itself is the TPE which allows extension of a trusted path to a distant, untrusted thin client and the related TCM which is capable of multiplexing multiple, single-level network connections onto a single interface. By

minimizing requirements for highly specialized hardware and software while adhering to sound security architecture, MYSEA aims to demonstrate the feasibility of general-purpose MLS environments.

## B. IPSEC OVERVIEW

IPsec is a standardized IP version 4 and version 6 protocol suite specified in RFC2401 [KEN98]. Its goal is to provide integrity, authentication, confidentiality, and replay protection for protocols operating at or above layer 3 in the ISO/OSI network model. By operating as a layer 3 protocol, TCP and UDP packets can be protected by the IPsec protocol. Additionally, by operating at layer 3, IPsec typically appears to be invisible to network-enabled application programs. Stated simply, IPsec works by encapsulating higher-level protocol packets within an IPsec packet much like TCP and UDP packets are encapsulated within IP packets.

As part of the suite, IPsec has two core protocols: authentication header (AH) and encapsulating security payload (ESP). The AH protocol provides authentication checking for entire IPsec packets while ESP provides for encryption of IPsec packets. ESP encapsulated packets are first encrypted and then enclosed within an IPsec packet with new IP header information. AH encapsulated packets are first enclosed within an IPsec packet with a new IP header and then a hash is computed over the *entire encapsulated* packet. It is interesting to note that neither protocol provides complete secrecy *and* integrity, as the AH protocol performs no encryption at all while packets containing ESP-encoded information are vulnerable to manipulation of their outer, IP-level headers. If both secrecy and integrity are functional concerns, IPsec is able to first encapsulate an IP packet using ESP for secrecy followed by an encapsulation of the ESP-encoded packet using AH. In some implementations this is not trivial, but by doing both, secrecy and integrity can be assured at the point of packet decryption.

IPsec's secrecy and integrity functionality is not tied to any single algorithm or set of algorithms for encryption and hashing. Instead, IPsec protection can be extended by adding additional hashing and encryption modules to an existing installation. Common hashing algorithms used in conjunction with IPsec include SHA1 and MD5, and common encryption algorithms are DES, 3DES, and AES. Since different hosts may have

multiple suites of algorithms that may be used to establish IPsec-protected communications, a method for dynamically negotiating the algorithms to be used in an IPsec flow is required.

Internet Key Exchange [HAR98] (IKE) was designed to be a protocol standard for negotiating and maintaining IPsec communications between hosts. It builds on the key exchange capabilities of the Oakley [ORM98] standard and the authentication and key exchange framework outlined in the Internet Security And Key Management Protocol [MAU98] (ISAKMP). When attempting to set up a secure session or "tunnel" between two systems or networks, IKE enables involved hosts to negotiate what services will be used, based on what protocols and algorithms each end host supports. These requirements may include a preferred or required hashing algorithm (e.g. SHA1, MD5) and/or a preferred or required encryption algorithm (e.g. DES, 3DES, AES). For example, if one host is only capable of using AES for encryption, while its peer offers a variety of secrecy algorithms, the IKE daemons on each host will attempt to negotiate a suite of protection algorithms common to both hosts that operates within the bounds of the policies of the system.. Upon successful negotiation of such algorithms, a security association (SA) is established for a given set of requirements. Any future communications that have the same requirements may also pass data under the same security association. Under the OpenBSD [OPE05] operating system, IKE functionality is provided by the isakmpd subsystem.

IPsec with IKE can provide some protection in a network environment, but by itself, it does not provide a mechanism for the specification of policies that might require conversations with a given host to use certain protocols while conversations with all other hosts must use a different set of protocols. Support for decision making and clear specification of policy can be found in a product called "KeyNote" [BLA99]. An IPsec-enabled implementation of KeyNote is included with OpenBSD versions 2.6 and greater [BLA01], and enables a system security administrator to specify policy rules that must be matched before a security association is permitted to take place between hosts.

At present, an IPsec implementation is not integrated with the IP stack for STOP. Until such an implementation is available, in order for the MYSEA server to leverage

IPsec-protected communications, implementation and testing must proceed with the insertion of an external, IPsec-capable device between the MLS server and the remote IPsec host or gateway. In the IPsec community, this is referred to as a "Bump In The Wire" (BITW) implementation.

## C. DYNAMIC SECURITY SERVICE/QUALITY OF SECURITY SERVICE

"Quality of Service" (QoS) in distributed computing and network environments typically describes attempts to quantify and maximize some weighted combination of factors such as efficiency, predictability, accuracy, and reliability across a set of given nodes, processes or entities as each entity attempts to complete some task. Sometimes it is used to provide ordering or prioritization of tasks or service requests. For example, in a corporate environment, a clerk might normally process requests based on the order they are received, regardless of what the request is for or whom it is from. If the clerk were to receive an executive's request marked "urgent", that clerk might *choose* to move the executive's request to the top of his to do list due a *policy* weighting executive requests marked urgent as being of the highest priority. We can say that the executive receives a different "quality of service" than everyone else the clerk serves, and possibly rightly so. Quality of Service can also involve requirements that a task be performed in a correct and/or timely manner or else it is not worth doing at all. For example, in grocery stores, produce is commonly discarded when it begins to look undesirable, despite its flavor or safety. The store simply has *chosen* that it is better to discard the food rather than sell something that looks unappealing or *might* taste substandard.

In "Quality of Security Service" [IRV00], the authors introduce the concept that in some environments, there is value in considering *security* as an additional dimension to QoS, coining the term "Quality of Security Service" in the process. For the purposes of this paper, the updated term "Dynamic Security Service" (DSS) will be used in place of QoSS. It is argued that security assurances, like requirements of accuracy and timeliness, can sometimes be either modulated within security contexts or traded on-and-off for other benefits such as system performance or data protection. In the spirit of the earlier examples, if a system were to receive a communication channel connection request to handle *sensitive but unclassified* data, the security subsystem might consider fulfilling the request for data transmission using only a moderate amount of cryptography if the

system's CPU were moderately loaded or might choose to employ a higher-degree of cryptography if the system's CPU were lightly loaded. This "floating" level of security service "quality" trades off secrecy and CPU availability. If however, the transmission of *top secret* data were to be required, the security subsystem might *require* that a stronger crypto algorithm be used, regardless of CPU availability or the resource-intensiveness of the operation. What we see is that security mechanisms can sometimes be implemented in a manner that does not necessarily result in negative trade-offs with other desirable conditions in a system, while at the same time providing that critical tasks be granted sufficient resources, regardless of the system's state.

Due to its flexible nature and many possibilities in implementation, IPsec is given as an example of an application that can be implemented within a DSS framework [SYP02a]. The IPsec protocol can be implemented using various forms of encryption and signing, each with a different potential impact on finite system resources such as network bandwidth or CPU-availability. This variability in functionality can also be played against factors such as external system threats in formulating a DSS policy that automatically, or at the push of a button, adapts to the changing security environment. A sample policy is represented by the following matrix:

| | | System Operational Mode | | |
|---|---|---|---|---|
| | | Normal | Crisis | Impacted |
| System Security Level | Low | AH: Integrity: MD5<br>ESP:Secrecy: DES<br>Integrity: MD5 | AH: Integrity:SHA1<br>ESP:Secrecy: 3DES<br>Integrity:SHA1 | AH: Integrity: MD5<br>ESP:Secrecy: DES<br>Integrity: MD5 |
| | Medium | AH: Integrity: MD5<br>ESP:Secrecy: CAST<br>Integrity:SHA1 | AH: Integrity:SHA1<br>ESP:Secrecy: 3DES<br>Integrity:SHA1 | AH: Integrity: MD5<br>ESP:Secrecy: DES<br>Integrity: MD5 |
| | High | AH: Integrity:SHA1<br>ESP:Secrecy: 3DES<br>Integrity:SHA1 | AH: Integrity:SHA1<br>ESP:Secrecy: AES<br>Integrity:SHA1 | AH: Integrity:SHA1<br>ESP:Secrecy: 3DES<br>Integrity: MD5 |

Table 1. Sample Dynamic Security Services Policy Database for IPsec

In this table, the current "threat level" faced by the system is represented by the column headings on the left, with "Low" indicating a low threat level and "High" indicating the system may be in a "lock-down" state while under attack. The "operational mode" dimension of the table might represent the availability of resources such as the total available CPU time of the system. For example, if a system is operating normally (Operational Mode = Normal) at a low threat level (Security level = Low) and a new threat on the network is identified, the security level may be switched to "High". In this case, the current IPsec security associations for AH-protected communications using the MD5 algorithm would be discarded. The IKE daemon would then consult its policy database and determine the appropriate algorithm to use for AH-operations to be "SHA". Through the key-exchange cycle, new security associations would be created between this host and its peers. Likewise, ESP-protected communications would stop using the security associations created with DES and MD5, and new associations would be created using 3DES and SHA. As covered in Chapter II.B, IPsec typically operates in a manner transparent to network applications and therefore, transparent to users as well. Hence, in the DSS framework, a change to a system's security level, operational mode, or both is to take place in a manner transparent to the end users and applications, hence communications would not be disrupted.

If the system were to now experience an event such as the loss of a processing node that would cause it to operate in an "impacted" state, as defined or determined by the system manager, the dynamic security policy would dictate that AH-protected communications continue to make use of the SHA algorithm, but that ESP-protected communications use a new algorithm pairing. The IKE daemon would be required to discard the security associations used for ESP-protected communications, consult its policy database, and create and use a new set of security associations using the 3DES and MD5 algorithms. Again, this shift in operational mode would take place without disrupting communications.

## D. INTEGRATING DSS INTO MYSEA

Traditionally, user interaction with an XTS-400 server takes place either at the console, over a directly-connected terminal, or over a single-level network interface directly from another trusted system such as a second XTS-400. In these situations, both

the user and the STOP assume the communication channel between the server and their interface is secure; viz. that communications will not be maliciously manipulated, dropped, or counterfeited. In today's highly-networked environments, the assurance requirements for a protected channel still remain, but the physical resource and distance restrictions imposed by terminal solutions and the like are impractical. Additionally, user-interface and cost issues add to this impracticality. These limitations are some of the driving factors that have led to the development of the TPE. As stated above, the TPE extends the trusted path out to a device that is separate from, yet co-located with, the stateless client PCs used for end-user interaction with the MYSEA server. Essentially, the TPE is a logical, as well as physical, extension of the MYSEA server. It is specified to respond to commands and application data from the MYSEA server, including acting as a router/NAT device while passing data on to the client PC. By its nature as a separate, evaluated device running an evaluated kernel, the TPE is able to extend the TCB to the boundary of the thin client. In the MYSEA environment, this extension of the trusted path is a necessary requirement.

In addition to providing trusted path and application data delivery services between the user and the MYSEA server, the TPE also provides what are termed "cryptography services" as it serves as one endpoint of an encrypted tunnel to the MYSEA server. The purpose for this cryptographic tunnel is both to protect the secrecy of data exchanged between the TPE (or the client via the TPE) and the server, and to authenticate communications between the TPE and the server. Authentication of traffic aides in making the TPE-server channel unforgable. By thus designing a mechanism for establishing secrecy and authentication between the TPE and the server, the network between the two endpoints need not be protected from eavesdropping or spoofing. Currently, implementation of these cryptographic services is being investigated using IPsec.

As cryptographic services are inherently part of the TPE-MYSEA server pairing, incorporating DSS to manage IPsec communications within this framework is a natural progression. As was previously covered in the background on DSS/QoSS, conditions on an operational network and on the nodes acting as endpoints on that network are often subject to change. Should the MYSEA server automatically detect or be explicitly

notified of a critical change in the status of its environment, the network carrying TPE communications, or relevant external conditions such as an increase in adversarial penetration attempts, it may be desirable that the MYSEA server be able to automatically raise the level of cryptographic or other security services between itself and any attached, subordinate TPEs. Following from the work outlined in [SYP02a] this thesis will further explore the integration of existing DSS proof-of-concept code into the MYSEA environment. From this, we now have a TPE that not only provides for a distributed TCB with a cryptographically-protected trusted path, but a TPE that is able to be dynamically reconfigured by is MLS server master as new requirements emerge under evolving operating conditions.

# III. REQUIREMENTS, DESIGN, AND IMPLEMENTATION

## A. REQUIREMENTS

The primary goal of this work is to integrate an existing IPsec-based DSS prototype [SYP02b] into the MYSEA environment. The two requirements of this integration effort work directly towards the implementation of two critical TPE services within the MYSEA environment: protected communication channels between the server and the TPE, and provision for DSS in managing these communication channels. [IRV04]

### 1. Protected Communication Channels

The MYSEA environment mandates provision for cryptographic secrecy and/or integrity protection of communications between the MYSEA server and the TPE, including client communications that pass through the TPE en route to the MYSEA server. While a fielded production environment may incorporate additional communication protection mechanisms including hardware encryption devices and the physical protection of networking equipment, this project implements and demonstrates protection of the following application protocols in the following manners:

#### *a. Protected Channel Protocol*

In this implementation, all communications using the Protected Channel Protocol across the MLS LAN are to be protected by IPsec operating in ESP mode. ESP has been selected for cryptographic secrecy as communications using the Protected Channel Protocol often contain sensitive authentication credentials (e.g. usernames, passwords, etc.).

#### *b. HTTP*

HyperText Transfer Protocol (HTTP) or "web" traffic traversing the MLS LAN is to be protected by IPsec operating in AH mode. HTTP has been chosen as the initial test protocol because web assess it the primary capability supported by MYSEA as well as is in common use in the DoD and commercial networks. The use of AH for HTTP and ESP for Protected Channel Protocol demonstrates the capability of the system to operate in both modes of IPsec operation at the same time. Upon deployment, HTTP

communications can be configured to operate under ESP and/or AH-protection following appropriate configuration of the security gateways.

Operation of both IPsec protocols under their respective protection mechanisms is demonstrated as part of this thesis research. Additionally, other application protocols and services can be added under protected communication channels by defining appropriate policies and flows for those services.

### 2. Dynamic Security Service

The second requirement of the MYSEA environment is to implement the protected communication channel services listed above in a manner that incorporates DSS. When fully implemented, the DSS policy will be dictated by the MYSEA server. The TPE/client side of the MLS LAN is to essentially operate as a "drone" that simply follows the DSS policy dictated by the MYSEA server. While a target of a complete DSS implementation in the MYSEA environment is to have the MLS server control the DSS security settings directly and dynamically, in this implementation it is sufficient that the server-side component of the DSS subsystem control the active security policy.

## B. DESIGN

This project has been broken-down into the following four stages, to reduce the implementation complexity of the project at large into a series of less-complex steps, each building on the former. Stages 1 and 2 are preliminary steps toward stages 3 and 4.

### 1. Implementation Stage 1


Figure 2. Stage 1 Design Topology

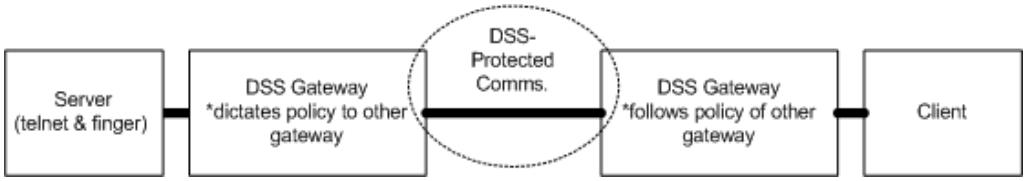


Figure 3. Stage 1 Logical Component Topology

Stage 1 reproduces former work in the DSS space [SYP02b] with some reconfiguration and extension. In this scenario, the OpenBSD DSS systems are configured to IPsec-protect the same protocols used in the former QoSS/DSS demonstrations, “finger” and “telnet”. Unlike the former work however, the DSS systems are extended to operate as IPsec security gateways, routing communications between a client and a server while at the same time protecting those communications as they pass over the shared MLS LAN. In this case, telnet and finger communications originate from the DSS client machine on the right, cross the MLS LAN with IPsec protection as mandated by the server-side gateway, are verified and, if necessary, decrypted by the server-side DSS gateway. Finally, the packet is delivered to the server. Server responses traverse the LAN in the reverse order and with the same protections across the MLS LAN.

The former DSS implementation only permitted communication to be initiated from the DSS system that dictated the DSS policies. In this stage however, the server-side DSS gateway maintains control over the current policy set, but the client gateway is permitted to initiate communication to or through the server gateway, as long as that communication conforms to the system security policy.

## 2. Implementation Stage 2

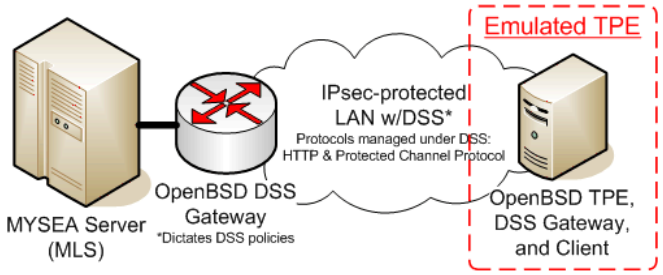


Figure 4. Stage 2 Design Topology


Figure 5. Stage 2, 3, & 4 Logical Component Topology

Stage 2 extends Stage 1 by replacing the generic server offering telnet and finger services with a MYSEA MLS server offering trusted path services and multilevel HTTP services. Again, as communications pass over the MLS LAN, they are cryptographically protected. Additionally, the configuration of the DSS subsystem is modified to protect HTTP and the Protected Channel Protocol, per requirement #1.

In this stage, the concept of a "Emulated TPE" is also introduced. The emulated TPE provides TPE services required by the MYSEA architecture, but may do so with additional, non-TPE-related functionality or may require more than one physical hardware or software component to provide such services. In this case, end-client functionality has been collapsed into the system serving as the TPE.

The TPE user interface functionality including secure attention key, MLS server login, secrecy/integrity level specification to the MLS server, etc. is provided through the used of a Java program previously developed by the MYSEA research team. Before the client-side system is permitted to make client-like application requests to the MYSEA server for services such as HTTP, the TPE software must be used to initiate a session with the MLS server.

### 3. Implementation Stage 3

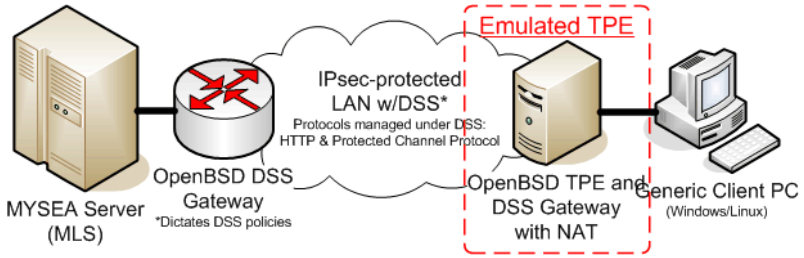


Figure 6. Stage 3 Design Topology

In Stage 3, a separate computer running a COTS operating system (Windows and/or Linux) is used as a MYSEA client. This configuration closely matches what is exhibited in the upper-right quadrant of Figure 1 – a client COTS PC communicating with a MLS server via a protected tunnel facilitated by the TPE. As stated in Chapter II, Section D, client communication through the TPE undergoes network address translation (NAT), making client communications appear to originate from the TPE from the server's perspective. The use of NAT adds complexity to the configuration of the DSS/TPE gateway. The protected protocols under DSS management continue to be HTTP and the Protected Channel Protocol.

Similar to Stage 2, the TPE user interface functionality (secure attention key, login, secrecy/integrity level specification, etc.) is provided by the same Java program described for Stage 2.

### 4. Implementation Stage 4

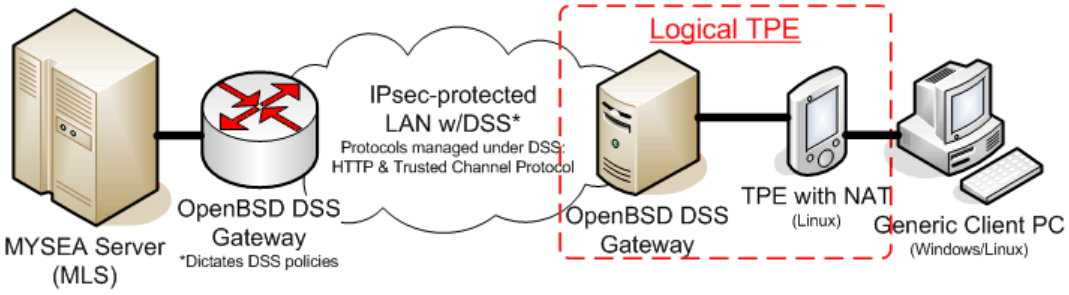


Figure 7. Stage 4 Design Topology

In the MYSEA environment, it is envisioned that the TPE device provides trusted path functionality, DSS, and NAT all on the same system. Since the current TPE prototype operating in a handheld form factor is not IPsec-aware, the DSS functionality for Stage 4 remained on a separate system.. In this scenario, the TPE running Linux on an iPAQ Pocket PC performs the user interface and NAT functions while the DSS gateway is responsible for communication protection and dynamic security services.

The protected protocols with DSS management are HTTP and the Protected Channel Protocol.

## C. IMPLEMENTATION DETAILS AND CHALLENGES

All 4 stages outlined in the design section have been successfully completed. This section outlines both implementation notes and difficulties faced in completing this project while working through each stage.

### 1. Physical Network Topology

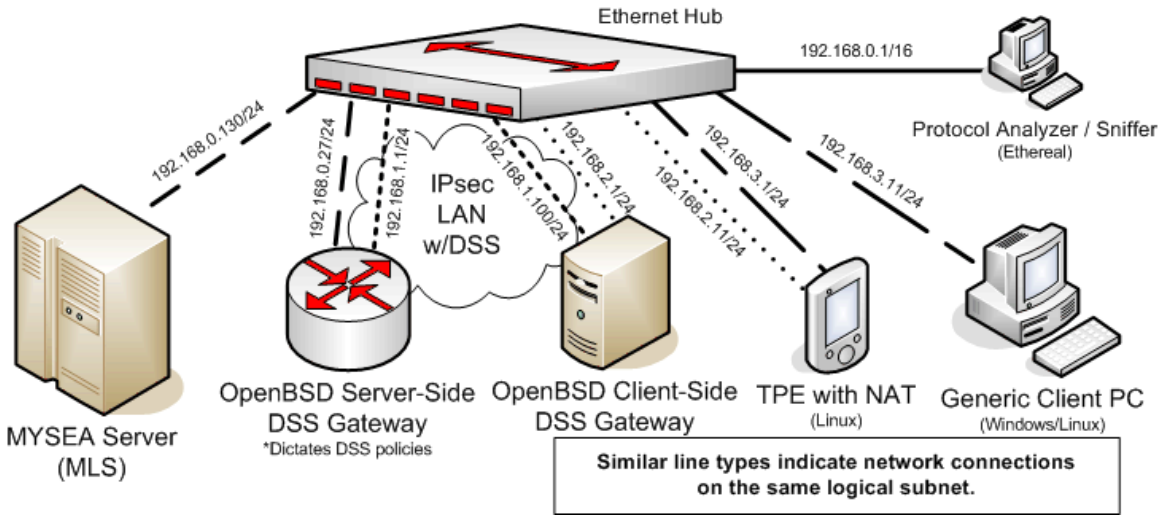


Figure 8. Physical Developmental Network Topology for Stage 4

When mapping the logical topologies of Stages 1 through 4 into physical implementations, the decision was made to perform all network and host interconnection via a single Ethernet hub. Through proper subnet definition on each participating network interface on each host, every network interface was restricted to communicate only with the hosts on its same logical network. During development, this created the

ability for a single host that is otherwise not a participant in the deployment to promiscuously monitor ("sniff") the traffic on all LAN segments, especially the MLS LAN. The ability to monitor and analyze traffic flows greatly aided troubleshooting when communications were not functioning correctly and enabled validation of correct communication flow and protection.

As a sample, the physical topology for Stage 4 is provided in Figure 8. Each interconnection line represents a host network interface with its address and subnet designation, and lines with similar textures or line types are on the same logical subnet. Upon deployment into the MYSEA test bed, the hub-based topology will be reconfigured to more closely match the logical topologies by not using a shared network medium.

### 2. Stages 1 and 2

From its early definition, DSS has remained an important concept in the MYSEA architecture, however while building on to and extending former MYSEA work in DSS, the availability of knowledge and experience with functional aspects of the preexisting DSS prototype was limited. Significant time was spent examining standard OpenBSD source code, DSS-modified OpenBSD source, and custom-written code in an effort to understand the system sufficiently well to extend it. While assembling background research on DSS and IPsec, the OpenBSD manual pages provided some assistance, but such help was generally limited, and in fact, the section in the OpenBSD FAQ dedicated to using IPsec has been off-line for months [OPE04].

Under OpenBSD, IPsec communications are typically configured in one of two ways: as statically defined and maintained channels using the *ipsecadm* utility or as dynamically negotiated and constructed channels using the isakmpd subsystem. The MYSEA DSS implementation of IPsec currently uses a hybrid of these two methods in its construction of a secure communication channel with dynamic properties. First, a "flow" must be defined from one DSS-enabled host to the other. The creation of IPsec flows is a normal part of a static OpenBSD IPsec deployment. This flow must be created on the DSS host that wishes to initiate protected communications with either the other DSS host or a system behind that host. In these implementations, flows are defined on the client-side DSS host for telnet, finger, HTTP, and the Protected Channel Protocol. The services are differentiated from one another based on their IP port numbers. In this

implementation, the flows are created by the script "/root/initialize_flows" which is called upon startup of the DSS system.

The second step in implementation is to configure the key management daemon. Unlike basic ipsecadm-configured IPsec communications, security associations are created and managed by the isakmpd subsystem. This involves configuring the file /etc/isakmpd/isakmpd.conf for phase-1 IPsec communications. A sample isakmpd.conf is included as an appendix. Once this file has been configured for a pair of DSS security gateways, it is not likely to need to be changed, even if additional flows are to be added. As the IKE daemon, isakmpd will use the IPsec flows created in the former step while creating and destroying security associations as needed.

The third step is to define the DSS KeyNote policy for communications protection between security gateways. This involves specification of which protocols are to be protected by ESP and/or AH and what cryptographic algorithms are to be used. This information is contained in the file /etc/isakmpd/isakmpd.policy on both DSS gateways.

**3. Stages 3 and 4**

The stage 3 implementation involves the addition of NAT functionality to the client-side DSS gateway as the client is moved to a physically separate machine. This created problems due to OpenBSD's limitations in NAT functionality and the operating system's mechanism used for identifying and routing packets containing Protected Channel Protocol communications. Protected Channel Protocol packets are identified by both DSS gateway subsystems for IPsec protection based on the value of their IP source port. It was discovered during implementation that if NAT is enabled under OpenBSD 3.0, the source port of every Protected Channel Protocol packet emerging from the combination NAT/DSS gateway is changed from 6033 to a random, high-numbered port. Due to this undesired manipulation of the IP source port, the KeyNote policies on the DSS gateways had to be changed to identify TPE Protected Channel traffic by the channel's use of the UDP protocol rather than by the IP source port 6033. As a side-effect, in a full, operational implementation with potentially dozens of protocols to be managed, *all* UDP-based application protocols, such as the domain name service (DNS), would fall under the same DSS policy as the Protected Channel Protocol. There may be relief for this issue in versions of OpenBSD greater than 3.0 that contain increased NAT

functionality, including the ability to perform NAT on packets leaving the client-side gateway while maintaining the packet's original source port. This would enable the KeyNote policy to again identify Protected Channel Protocol communications based on a port specification rather than based on the UDP protocol alone.

Another NAT-related issue was encountered in Stage 3. IPsec rules must be specified on the client-side DSS subsystem with the real addresses of both the client and the MLS server. Upon client-side initiation of communications with the MYSEA server, the server-side DSS gateway creates matching IPsec rules from the MLS server to the client by essentially reversing the properties of the client-initiated rules. However, due to NAT, the MLS server and the server-side DSS gateway are not able to directly communicate with the client computer as server-to-client communications must be proxied by the DSS/NAT gateway. For communications from the MLS server to receive protection, IPsec rules are required to be explicitly defined in the *vpn28_ah_a* script (appendix C, section A.1.a) on the server-side DSS gateway, specifying the destination as the combined TPE/DSS gateway rather than the actual client computer. The extension of Stage 3 to Stage 4 brought relief for both NAT issues as all NAT functionality was moved to the external TPE.

During the implementation and testing of Stages 3 and 4, it was discovered that following a dynamic policy reconfiguration by the server-side gateway, the next TPE-initiated Protected Channel Protocol UDP packet triggers the creation of a new set of security associations as expected, but that request is then dropped or "lost" by the client-side IPsec gateway. It is possible that the first packet of each protocol protected by the DSS IPsec gateway is dropped if security associations for that protocol do not already exist, and that TCP-based application protocols such as HTTP are tolerant of the loss and automatically retry their initial request. The current workaround for this issue is to require the user to press the SAK on the TPE a second time if the first SAK did not produce a server response. This workaround is not acceptable in a final DSS-enabled IPsec implementation in the MYSEA testbed.

With the completion of Stages 3 and 4, two implementations of a set of protected communications channels incorporating dynamic security services have been completed.

By building on an existing DSS prototype and extending it using a stepwise approach, the goals of creating protected communication channels for different protocols used in the MYSEA environment, each operating under different dynamic security service policies, was met. Unit testing for correct functioning of the system is covered in Chapter IV, and comments on possible future work, including further integration into the MYSEA testbed are included in Chapter V.

# IV. UNIT TESTING

When implementing a new system or extending an existing one, testing is necessary to ensure the system complies with the requirements laid out prior to the design of the system. Testing may uncover implementation and configuration errors, design deficiencies, and/or unexpected functionality of the system. This test plan uses both exhaustive and selective techniques to examine if the IPsec DSS implementation meets the requirements specified in Chapter III.

## A. TEST PLAN

The purpose of the unit test plan is to validate the system's implementation against the requirements laid out previously. Since both Stages 3 and 4 must be tested, all of the following tests must be repeated in both environments unless otherwise noted. The topology and setup for the tests are the same as those used for the demonstrations of Stages 3 and 4 and are found in Appendicies A and B. The test procedures are found in Appendix D.

### 1. Protection of Data Crossing the MLS LAN

The purpose of this test is to verify that communications purported to be protected by IPsec services on the MLS LAN are, in fact, protected in a manner consistent with the configured policy. While former work in Quality of Security Service at the Naval Postgraduate School has involved some testing of the DSS (then-called QoSS) subsystem for correct functionality in a point-to-point implementation, the extension of DSS-enabled endpoints into security gateways not only requires testing of new functionality, but also the retesting of preexisting functionality.

The test plan is presented in Table 2. The System Mode and System Level indicate the current policy the server gateway will dictate. The Protocol Protected indicates the client or TPE transactions that fall under the policy being tested and are either "Protected Channel" or "HTTP". The IPsec Protocol column indicates whether AH or ESP protections should be applied to the protected protocol, and the Expected Algorithm(s) column indicates which protection suite should be used by the IPsec protocol under test.

| Test Number | System Mode | System Level | Protocol Protected | IPsec Protocol | Expected Algorithm(s) on MLS LAN |
|---|---|---|---|---|---|
| a1 | default | default | Protected Channel | ESP | DES/MD5 |
| a2 | default | default | HTTP | AH | MD5 |
| a3 | normal | low | Protected Channel | ESP | DES/MD5 |
| a4 | normal | low | HTTP | AH | MD5 |
| a5 | normal | medium | Protected Channel | ESP | CAST/SHA |
| a6 | normal | medium | HTTP | AH | MD5 |
| a7 | normal | high | Protected Channel | ESP | 3DES/SHA |
| a8 | normal | high | HTTP | AH | SHA |
| a9 | crisis | low | Protected Channel | ESP | 3DES/SHA |
| a10 | crisis | low | HTTP | AH | SHA |
| a11 | crisis | medium | Protected Channel | ESP | 3DES/SHA |
| a12 | crisis | medium | HTTP | AH | SHA |
| a13 | crisis | high | Protected Channel | ESP | AES/SHA |
| a14 | crisis | high | HTTP | AH | SHA |
| a15 | impacted | low | Protected Channel | ESP | DES/MD5 |
| a16 | impacted | low | HTTP | AH | MD5 |
| a17 | impacted | medium | Protected Channel | ESP | DES/MD5 |
| a18 | impacted | medium | HTTP | AH | MD5 |
| a19 | impacted | high | Protected Channel | ESP | 3DES/MD5 |
| a20 | impacted | high | HTTP | AH | SHA |

Table 2. MLS LAN Data Protection Test

NOTE: “SHA” and “MD5” in Table 2 are abbreviations for HMAC-SHA1 and HMAC-MD5 implementations of IPsec message authenticity protection techniques.

**2. Dynamic Policy Tear-Down**

As the DSS gateways are being tested for properly following the current security service policy, the *dynamic* nature of the system’s security associations must be tested. Not only must the new algorithm used for protection be negotiated when there is a change in the current DSS policy, but any SA’s created with the old algorithm must be destroyed. The test cases described in Table 3 are based on the MLS LAN Data Protection Test Suite. The test conditions column represents the test condition transitions between two test scenarios. The last three columns show the expected results of a change in security association and dynamic policy flows.

| Test Number | System Change Conditions (See Table 2 for details) | Security Association Properly Destroyed Upon Policy Change: | | Expected Status of Dynamic Policy Flows on Server Gateway |
|---|---|---|---|---|
| | | Expected Server Gateway Result | Expected Client Gateway Result | |
| b1 | a2 to a3 | SA’s destroyed | SA’s destroyed | Deleted |
| b2 | a3 to a4 | SA’s destroyed | SA’s destroyed | Deleted |
| b3 | a4 to a5 | SA’s destroyed | SA’s destroyed | Deleted |
| b4 | a14 to a15 | SA’s destroyed | SA’s destroyed | Deleted |
| b5 | a18 to a19 | SA’s destroyed | SA’s destroyed | Deleted |
| b6 | a19 to a20 | SA’s destroyed | SA’s destroyed | Deleted |

Table 3. Selective Dynamic Policy Tear-Down Test

### 3. Correct NAT Functionality

This test suite provides assurance that the DSS subsystem can properly handle NAT'ed traffic. Network packet analysis for this test can be performed using an additional computer attached to the network hub in a manner that allows it to view traffic flows. Optionally, the traffic analysis can be performed on the server-side DSS gateway running Ethereal software.

| Test Number | Client Protocol/Traffic Type | Expected Client Address On MLS LAN |
|---|---|---|
| c1 | Protected Channel Protocol | Stage 3: 192.168.1.100 |
| | | Stage 4: 192.168.2.11 |
| c2 | HTTP | Stage 3: 192.168.1.100 |
| | | Stage 4: 192.168.2.11 |

Table 4. NAT Functionality Test

Table 4 shows the expected values of the source and destination IP addresses for both client-to-server requests and server-to-client responses for Protected Channel Protocol and HTTP as the requests and responses cross the MLS LAN. The addresses correspond to the MLS LAN-side address of the NAT system.

## B. TEST REPORT

This section provides the results of the testing of Stages 3 and 4 of the DSS-enabled IPsec implementation. Tables 2 through 4 have been replicated below, and an additional column added to each indicating the results of the test.

### 1. Stage 3 Testing

The following test results are the result of testing in the Stage 3 environment.

#### *a. MLS LAN Data Protection Test*

This test demonstrates that the DSS IPsec subsystem uses the correct protection algorithms in accordance with the system security policy.

| Test Number | System Mode | System Level | Protocol Protected | IPsec Protocol | Expected Algorithm(s) on MLS LAN | Observed Algorithm(s) on MLS LAN |
|---|---|---|---|---|---|---|
| a1 | default | default | Protected Chnl. | ESP | DES/MD5 | DES/MD5 |
| a2 | default | default | HTTP | AH | MD5 | MD5 |
| a3 | normal | low | Protected Chnl. | ESP | DES/MD5 | DES/MD5 |
| a4 | normal | low | HTTP | AH | MD5 | MD5 |
| a5 | normal | medium | Protected Chnl. | ESP | CAST/SHA | CAST/SHA |
| a6 | normal | medium | HTTP | AH | MD5 | MD5 |
| a7 | normal | high | Protected Chnl. | ESP | 3DES/SHA | 3DES/SHA |
| a8 | normal | high | HTTP | AH | SHA | SHA |
| a9 | crisis | low | Protected Chnl. | ESP | 3DES/SHA | 3DES/SHA |
| a10 | crisis | low | HTTP | AH | SHA | SHA |
| a11 | crisis | medium | Protected Chnl. | ESP | 3DES/SHA | 3DES/SHA |
| a12 | crisis | medium | HTTP | AH | SHA | SHA |
| a13 | crisis | high | Protected Chnl. | ESP | AES/SHA | AES/SHA |
| a14 | crisis | high | HTTP | AH | SHA | SHA |
| a15 | impacted | low | Protected Chnl. | ESP | DES/MD5 | DES/MD5 |
| a16 | impacted | low | HTTP | AH | MD5 | MD5 |
| a17 | impacted | medium | Protected Chnl. | ESP | DES/MD5 | DES/MD5 |
| a18 | impacted | medium | HTTP | AH | MD5 | MD5 |
| a19 | impacted | high | Protected Chnl. | ESP | 3DES/MD5 | 3DES/MD5 |
| a20 | impacted | high | HTTP | AH | SHA | SHA |

Table 5. Stage 3 MLS LAN Data Protection Test Results

### *b. Selective Dynamic Policy Tear-Down Test*

This test demonstrates that the DSS subsystem correctly destroys security associations upon a transition between any two given test cases from the MLS LAN Data Protection Test.

| Test Number | System Change Conditions (See Table 2 for details) | Security Association Properly Destroyed Upon Policy Change: | | Expected Status of Dynamic Policy Flows on Server Gateway | Observed Status of Dynamic Policy Flows on Server Gateway |
|---|---|---|---|---|---|
| | | Expected Server Gateway Result | Expected Client Gateway Result | | |
| b1 | a2 to a3 | SA's destroyed | SA's destroyed | Deleted | Deleted |
| b2 | a3 to a4 | SA's destroyed | SA's destroyed | Deleted | Deleted |
| b3 | a4 to a5 | SA's destroyed | SA's destroyed | Deleted | Deleted |
| b4 | a14 to a15 | SA's destroyed | SA's destroyed | Deleted | Deleted |
| b5 | a18 to a19 | SA's destroyed | SA's destroyed | Deleted | Deleted |
| b6 | a19 to a20 | SA's destroyed | SA's destroyed | Deleted | Deleted |

Table 6. Stage 3 Selective Dynamic Policy Tear-Down Test Results

### *c. NAT Functionality Test*

This test demonstrates the DSS subsystem can properly handle NAT'ed traffic. Network packet analysis for this test was performed using a computer attached to the network hub in a manner that allows it to view all traffic flows.

| Test Number | Client Protocol/Traffic Type | Expected Client Address On MLS LAN | Observed Client Address On MLS LAN |
|---|---|---|---|
| c1 | Protected Channel Protocol | 192.168.1.100 | 192.168.1.100 |
| c2 | HTTP | 192.168.1.100 | 192.168.1.100 |

Table 7. Stage 3 NAT Functionality Test Results

### 2. Stage 4 Testing

The following test results are the result of testing in the Stage 4 environment.

#### *a. MLS LAN Data Protection Test*

This test demonstrates that the DSS subsystem uses the correct protection algorithms in accordance with the system security policy.

| Test Number | System Mode | System Level | Protocol Protected | IPsec Protocol | Expected Algorithm(s) on MLS LAN | Observed Algorithm(s) on MLS LAN |
|---|---|---|---|---|---|---|
| a1 | default | default | Protected Chnl. | ESP | DES/MD5 | DES/MD5 |
| a2 | default | default | HTTP | AH | MD5 | MD5 |
| a3 | normal | low | Protected Chnl. | ESP | DES/MD5 | DES/MD5 |
| a4 | normal | low | HTTP | AH | MD5 | MD5 |
| a5 | normal | medium | Protected Chnl. | ESP | CAST/SHA | CAST/SHA |
| a6 | normal | medium | HTTP | AH | MD5 | MD5 |
| a7 | normal | high | Protected Chnl. | ESP | 3DES/SHA | 3DES/SHA |
| a8 | normal | high | HTTP | AH | SHA | SHA |
| a9 | crisis | low | Protected Chnl. | ESP | 3DES/SHA | 3DES/SHA |
| a10 | crisis | low | HTTP | AH | SHA | SHA |
| a11 | crisis | medium | Protected Chnl. | ESP | 3DES/SHA | 3DES/SHA |
| a12 | crisis | medium | HTTP | AH | SHA | SHA |
| a13 | crisis | high | Protected Chnl. | ESP | AES/SHA | AES/SHA |
| a14 | crisis | high | HTTP | AH | SHA | SHA |
| a15 | impacted | low | Protected Chnl. | ESP | DES/MD5 | DES/MD5 |
| a16 | impacted | low | HTTP | AH | MD5 | MD5 |
| a17 | impacted | medium | Protected Chnl. | ESP | DES/MD5 | DES/MD5 |
| a18 | impacted | medium | HTTP | AH | MD5 | MD5 |
| a19 | impacted | high | Protected Chnl. | ESP | 3DES/MD5 | 3DES/MD5 |
| a20 | impacted | high | HTTP | AH | SHA | SHA |

Table 8. Stage 4 MLS LAN Data Protection Test Results

### *b. Selective Dynamic Policy Tear-Down Test*

This test demonstrates that the DSS IPsec subsystem correctly destroys security associations upon a transition between any two given test cases from the MLS LAN Data Protection Test.

| Test Number | System Change Conditions (See Table 2 for details) | Security Association Properly Destroyed Upon Policy Change: | | Expected Status of Dynamic Policy Flows on Server Gateway | Observed Status of Dynamic Policy Flows on Server Gateway |
|---|---|---|---|---|---|
| | | Expected Server Gateway Result | Expected Client Gateway Result | | |
| b1 | a2 to a3 | SA's destroyed | SA's destroyed | Deleted | Deleted |
| b2 | a3 to a4 | SA's destroyed | SA's destroyed | Deleted | Deleted |
| b3 | a4 to a5 | SA's destroyed | SA's destroyed | Deleted | Deleted |
| b4 | a14 to a15 | SA's destroyed | SA's destroyed | Deleted | Deleted |
| b5 | a18 to a19 | SA's destroyed | SA's destroyed | Deleted | Deleted |
| b6 | a19 to a20 | SA's destroyed | SA's destroyed | Deleted | Deleted |

Table 9. Stage 4 Selective Dynamic Policy Tear-Down Test Results

### *c. NAT Functionality Test*

This test demonstrates the DSS subsystem can properly handle NAT'ed traffic. Network packet analysis for this test was performed using a computer attached to the network hub in a manner that allows it to view all traffic flows.

| Test Number | Client Protocol/Traffic Type | Expected Client Address On MLS LAN | Observed Client Address On MLS LAN |
|---|---|---|---|
| c1 | Protected Channel Protocol | 192.168.2.11 | 192.168.2.11 |
| c2 | HTTP | 192.168.2.11 | 192.168.2.11 |

Table 10. Stage 4 NAT Functionality Test Results

# V. FUTURE WORK AND CONCLUSIONS

## A. FUTURE WORK

The MYSEA architecture is constantly undergoing development, extension, and refinement, and as part of this environment, it is expected the IPsec-DSS subsystem will experience similar evolution. The following issues have been identified as follow-on work that will advance the MYSEA project as well.

### 1. Multi-user/Multi-TPE Integration and Testing

An immediate follow-on project to this work is to integrate multiple client-side DSS gateways, TPEs, and clients with the current server-side DSS gateway and MLS server. Integration is expected to be relatively straightforward with only changes required in the configuration of the isakmpd daemon and the definition of additional DSS rule pairings on the server-side gateway and the client-side gateways. Additionally, testing of multiple simultaneous sessions with different IPsec/IKE policies would be required to ensure communications between a given TPE and the MLS server are correctly delivered to that TPE and are afforded appropriate integrity and secrecy protections.

### 2. MLS Server Control Over Dynamic Security Policy

In the MYSEA environment, policy and access control for all components, including the DSS subsystem, is to be controlled by the MYSEA MLS server. Since the security policies are currently manually entered into the DSS security gateways, an automated mechanism should be implemented for the MLS server to contact the server-side DSS gateway, change its current policy attributes, and trigger isakmpd to notify the client-side DSS gateway of the policy change. In the current system architecture, this can be accomplished by establishing a connection to the server gateway, updating or creating a new dynamic policy file, and forcing a reconfiguration and restart of isakmpd via the daemon's filesystem FIFO interface.

### 3. Integration of the DSS Gateways into the MLS Server and TPE

To further integrate the DSS subsystem into the MYSEA architecture, the functionality of the DSS gateways should be ported from the OpenBSD systems onto the MYSEA MLS server and the TPE prototype. At a minimum, both integration targets

must support IPsec, and ideally, both IPsec implementations would be fully compatible with isakmpd's operation. Since an IPsec implementation for the XTS-400's STOP is not available at this time and IPsec support for the Pocket PC-based TPE is limited, this work may require a significant investment of time and manpower to realize.

#### 4. Mechanism for MLS Server to Update TPE

Another project that is important to a complete implementation of secure server-to-TPE communications with DSS is the ability of the MLS server to not only dictate the security policies the TPE must follow, but also remotely update the TPE in a secure manner with new policy and flow definitions, updated packet filtering and NAT rules, and other system-level changes such as software patches.

## B. CONCLUSIONS

This project has produced a dynamic security service-enabled IPsec subsystem for the MYSEA environment. This implementation has afforded adaptable secrecy and integrity protection for both sensitive authentication and integrity-critical information delivery tasks in the MYSEA environment. By also providing a dynamic resource-aware dimension to the protection of data communications, MYSEA operating policies can be constructed in a manner that makes optimal use of finite system resources in the execution of security functions. It is envisioned that as threats are introduced into the MYSEA testbed and system resources are made available or are consumed by the completion or execution of tasks, the MYSEA trusted components will be able to dynamically adjust its operational security policies in a manner that provides for business continuity while maintaining high assurance.

In a broader scope, the concepts demonstrated in this system can aid the development of the Global Information Grid by providing flexible, policy-aware secrecy and integrity assurance. The underlying goals of the MYSEA project remain focused on developing, refining, and integrating practices, products, and services that closely match the requirements of the GIG. As the GIG is extended to mesh the networks, users, and missions of DoD, law enforcement, and intelligence communities, the concepts of dynamic security services for the protection of network communications become essential.

# APPENDIX A: STAGE 3 SYSTEM INSTALLATION AND DEMONSTRATION

These instructions describe how to setup and execute a demonstration of the DSS system in a Stage 3 implementation. In this scenario, client-server HTTP communication is facilitated through the use of a software TPE for authentication services. The DSS IPsec subsystem is used to provide ESP cryptographic secrecy and integrity protection for the authentication protocol and AH integrity protection for the HTTP protocol.

Included in this demonstration is normal TPE functionality including login, session level negotiation, and session level renegotiation combined with DSS policy modulation. It is shown that the DSS implementation unobtrusively augments normal MYSEA LAN functionality while providing TPE authentication services, NAT functionality, and DSS on the same physical system.

A. Network Topology

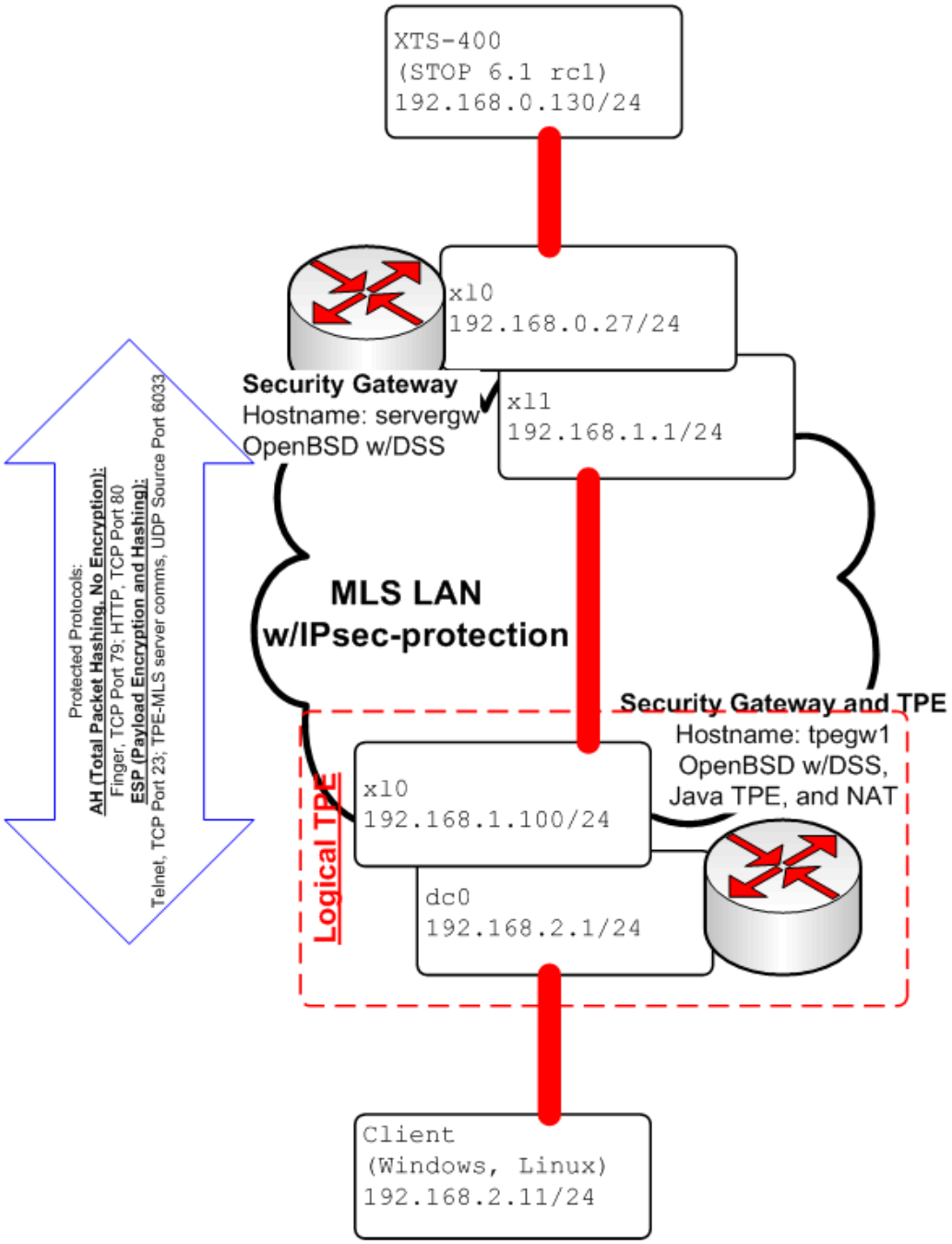


Figure 9. Stage 3 Logical Network Topology

B. Equipment Requirements

B.1. XTS-400

B.1.1. MYSEA environment installed and configured

B.1.2. Set the default route for MLS LAN tcpip daemon to the MLS LAN address of the server gateway (servergw, 192.168.0.27)

B.1.3. MLS LAN IP address of TPE (192.168.1.100) must be listed in `/usr/local/mysea/tcbe_list`

B.1.4. The TPE-facing address of the MYSEA server has been configured to 192.168.0.130

B.2. Combination Security Gateway and TPE

B.2.1. Intel x86 Pentium-class machine or better

B.2.2. Video card supported under XFree86 4.1.0

B.2.3. Two (2) OpenBSD 3.0-supported network interfaces

B.3. Server-Side Gateway System

B.3.1. Intel x86 Pentium-class machine or better

B.3.2. Video card supported under XFree86 4.1.0

B.3.3. Two (2) OpenBSD 3.0-supported network interfaces

B.4. Additional equipment

B.4.1. Hubs, switches, cables, and/or cross-over cables sufficient to implement the network architecture pictured above

B.4.2. OpenBSD 3.0 install media.

B.4.3. MYSEA Dynamic Security Service install media (CD-ROM)

C. Installation and Configuration

C.1. TPE Gateway System

C.1.1. Install OpenBSD 3.0 from CD

C.1.2. If OpenBSD is preinstalled and has its networking configured, the following changes may be necessary:

C.1.2.1. Modify `/etc/hosts` to include correct IP address-hostname pairings for this system, per the instructions below and the architecture depicted above

C.1.2.2. Modify `/etc/myname` to contain the correct hostname of the system

C.1.2.3. Modify `/etc/hostname.IF_NAME` with correct IP address information where "`IF_NAME`" is the name of each network interface on the system

C.1.2.4. Modify `/etc/mygate` to contain the system's default gateway

C.1.2.5. The remaining OS install instructions can be skipped

C.1.3. Configure hard disk to have one partition (the 'a' partition) with most of the disk space, leaving between 200-500 meg for the swap partition.

C.1.4. Configure:

System name: `tpegw1`

No domain

IP address of the MLS-LAN facing interface: `192.168.1.100`

Netmask: `255.255.255.0`

Default route: `192.168.1.1`

Primary nameserver: ‘none’

IP address of the client-facing interface: `192.168.2.1`

Netmask `255.255.255.0`

No default route

C.1.5. See the MLS LAN administrator for the password to use for the ‘root’ user.

C.1.6. Install all sets from the CD.

C.1.7. After install, if the system will not boot, rewrite the master boot record:

C.1.7.1. Boot using a DOS floppy or Windows 98 install CD containing fdisk

C.1.7.2. Run `fdisk /mbr`

C.1.8. Other system configuration

C.1.8.1. Configure X-Windows

C.1.8.2. Configure X-Windows to start automatically:

C.1.8.3. Edit `/etc/rc.conf`

change: `xdm_flags=NO`

to: `xdm_flags=””`

C.1.8.4. Configure NAT (Network Address Translation)

C.1.8.4.1. Enable packet forwarding

Edit `/etc/sysctl.conf`

change: `#net.inet.ip.forwarding=1`

to: `net.inet.ip.forwarding=1`

C.1.8.4.2. Enable firewall (PF) and NAT

Edit `/etc/rc.conf`

change: `pf=NO`

to: `pf=YES`

C.1.8.4.3. Edit NAT configuration

Edit `/etc/nat.conf`, adding the following line to the bottom of the file:

```
nat on enc0 from 192.168.2.0/24 to
192.168.0.0/24 -> 192.168.1.100
```

A sample `/etc/nat.conf` can be found on the DSS Install CD in the `/tpegw/etc` directory.

C.1.8.4.4. The NAT-changes will be applied upon the next system reboot.

C.1.8.5. Add the following lines to `/etc/fstab`

```
/dev/fd0a /mnt/floppy    msdos  rw,-l,noauto 0    0
/dev/cd0a /mnt/cdrom     cd9660 ro,noauto    0    0
/kern     /kern          kernfs ro           0    0
```

C.1.8.6. Create the mount directories

C.1.8.6.1. `mkdir /mnt/floppy`

C.1.8.6.2. `mkdir /mnt/cdrom`

C.1.8.6.3. `mkdir /kern`

C.1.9. Install Kaffe (Java)

C.1.9.1. Mount the DSS Install CD: `mount /mnt/cdrom`

C.1.9.2. Add the Kaffe package

```
pkg_add –v /mnt/cdrom/packages/kaffe-1.0.6.tgz
```

It may be necessary to install other packages Kaffe depends on. These packages are located in the same directory as the Kaffe package.

C.1.9.3. Edit the configuration for added shared libraries

Add the following line to /etc/rc.conf:

```
shlib_dirs=”/usr/local/bin/kaffe”
```

C.1.9.4. Run the script which installs ‘swing.jar’

```
cd /mnt/cdrom/isakmp_mon_responder
./inst
```

C.1.10. Reboot.

C.2. Server-Side Gateway System

C.2.1. Install OpenBSD 3.0 from CD

C.2.2. If OpenBSD is preinstalled and has its networking configured, the following changes may be necessary:

C.2.2.1. Modify `/etc/hosts` to include correct IP address-hostname pairings for this system, per the instructions below and the architecture depicted above

C.2.2.2. Modify `/etc/myname` to contain the correct hostname of the system

C.2.2.3. Modify `/etc/hostname.IF_NAME` with correct IP address information where "`IF_NAME`" is the name of each network interface on the system

C.2.2.4. Modify `/etc/mygate` to contain the system's default gateway

C.2.2.5. The remaining OS install instructions can be skipped

C.2.3. Configure hard disk to have one partition (the 'a' partition) with most of the disk space, leaving between 200-500 meg for the swap partition.

C.2.4. Configure:

System name: `servergw`

No domain

IP address of the MLS-LAN facing interface: `192.168.1.1`

Netmask: `255.255.255.0`

Default route: `192.168.1.100`

Primary nameserver: 'none'

IP address of the MLS-server-facing interface: `192.168.0.27`

Netmask `255.255.255.0`

No default route

C.2.5. See the MLS LAN administrator for the password to use for the 'root' user

C.2.6. Install all sets from the CD.

C.2.7. After install, if the system will not boot, rewrite the master boot record:

C.2.7.1. Boot using a DOS floppy or Windows 98 install CD containing fdisk

C.2.7.2. Run `fdisk /mbr`

C.2.8. Other system configuration

C.2.8.1. Configure X-Windows

C.2.8.2. Configure X-Windows to start automatically:

C.2.8.3. Edit `/etc/rc.conf`

change: `xdm_flags=NO`

to: `xdm_flags=""`

C.2.8.4. Configure NAT (Network Address Translation)

C.2.8.4.1. Enable packet forwarding

Edit `/etc/sysctl.conf`

change: `#net.inet.ip.forwarding=1`

to: `net.inet.ip.forwarding=1`

C.2.8.4.2. Enable firewall (PF) and NAT

Edit `/etc/rc.conf`

change: `pf=NO`

to: `pf=YES`

C.2.8.5. Add the following lines to `/etc/fstab`

```
/dev/fd0a /mnt/floppy    msdos  rw,-l,noauto 0    0
/dev/cd0a /mnt/cdrom     cd9660 ro,noauto    0    0
/kern     /kern          kernfs ro           0    0
```

C.2.8.6. Create the mount directories

C.2.8.6.1. `mkdir /mnt/floppy`

C.2.8.6.2. `mkdir /mnt/cdrom`

C.2.8.6.3. `mkdir /kern`

C.2.9. Install Kaffe (Java)

C.2.9.1. Mount the DSS Install CD: `mount /mnt/cdrom`

C.2.9.2. Add the Kaffe package

```
pkg_add –v /mnt/cdrom/packages/kaffe-1.0.6.tgz
```

It may be necessary to install other packages Kaffe depends on. These packages are located in the same directory as the Kaffe package.

C.2.9.3. Edit the configuration for added shared libraries

Add the following line to /etc/rc.conf:

```
shlib_dirs=”/usr/local/bin/kaffe”
```

C.2.9.4. Run the script which installs ‘swing.jar’

C.2.9.4.1. `cd /mnt/cdrom/isakmp_mon_responder`

C.2.9.4.2. `./inst`

C.2.10. Reboot

C.3. DSS Changes on Combination Security Gateway and TPE

C.3.1. *Optional* – Install Ethereal (network packet analyzer)

C.3.1.1. Mount the OpenBSD 3.0 CD1

C.3.1.2. Add the Ethereal package (and any other packages it depends on) with the command:

```
pkg_add –v /mnt/cdrom/3.0/packages/i386/ethereal-0.8.19.tgz
```

C.3.1.3. Edit the configuration for added shared libraries

Add the following line to /etc/rc.conf:

```
shlib_dirs=”/usr/local/bin/kaffe /usr/local/bin/pth”
```

C.3.2. Install kernel and isakmpd code changes

C.3.2.1. Install the OpenBSD 3.0 source

C.3.2.1.1. Mount the OpenBSD 3.0 CD3

C.3.2.1.2. `cd /usr/src`

C.3.2.1.3. `tar zxvf /mnt/cdrom/src.tar.gz`

C.3.2.2. Install updated system files

C.3.2.2.1. Mount the DSS Install CD

C.3.2.2.2. `cd /tmp`

C.3.2.2.3. `tar zxvf /mnt/cdrom/cvs.tar.gz`

C.3.2.3. Copy the changed files into `/usr/src`

C.3.2.3.1. `cd /tmp/src/sbin/isakmpd`

C.3.2.3.2. `cp ipsec.c pf_key_v2.c sa.h sa.c ui.c /usr/src/sbin/isakmpd`

C.3.2.3.3. `cp /tmp/src/sys/net/pfkeyv2.c /usr/src/sys/net`

C.3.2.3.4. `cp /tmp/src/sys/netinet/ip_spd.c /usr/src/sys/netinet`

C.3.2.4. Rebuild the kernel

C.3.2.4.1. `cd /usr/src/sys/arch/i386/conf`

C.3.2.4.2. `config GENERIC`

C.3.2.4.3. `cd ../compile/GENERIC`

C.3.2.4.4. `make depend ; make`

C.3.2.4.5. `mv /bsd /bsd.orig`

C.3.2.4.6. `cp bsd /bsd`

C.3.2.4.7. `Rebuild isakmpd`

C.3.2.4.8. `cd /usr/src/sbin/isakmpd`

C.3.2.4.9. `make obj ; make depend ; make ; make install`

C.3.2.5. Setup isakmpd configuration files

C.3.2.5.1. Mount the DSS Install CD

C.3.2.5.2. `cd /mnt/cdrom/tpegw/etc/isakmpd`

C.3.2.5.3. `cp isakmpd.conf /etc/isakmpd/`

C.3.2.5.4. `cp isakmpd.policy* /etc/isakmpd/`

C.3.2.5.5. `cd /etc/isakmpd`

C.3.2.5.6. `chmod 600 isakmpd.conf`

C.3.2.5.7. `chmod 600 isakmpd.policy*`

C.3.2.5.8. `mv isakmpd.policy.NAT isakmpd.policy`

C.3.2.6. Install policy setup, Java TPE, and DSS GUI files

C.3.2.6.1. `cd /mnt/cdrom/tpegw/root`

C.3.2.6.2. `cp –R ./* /root`

C.3.2.6.3. `cd /root`

C.3.2.6.4. `mv initialize_flows.NAT initialize_flows`

C.3.2.7. Test the java installation by executing `/root/tpe`

C.4. DSS Changes on Server-Side Gateway System

C.4.1. *Optional* – Install Ethereal (network packet analyzer)

C.4.1.1. Mount the OpenBSD 3.0 CD1

C.4.1.2. Add the Ethereal package (and any other packages it depends on) with the command:

```
pkg_add –v /mnt/cdrom/3.0/packages/i386/ethereal-0.8.19.tgz
```

C.4.1.3. Edit the configuration for added shared libraries

Add the following line to /etc/rc.conf:

```
shlib_dirs=”/usr/local/bin/kaffe /usr/local/bin/pth”
```

C.4.2. Install kernel and isakmpd code changes

C.4.2.1. Install the OpenBSD 3.0 source

C.4.2.1.1. Mount the OpenBSD 3.0 CD3

C.4.2.1.2. `cd /usr/src`

C.4.2.1.3. `tar zxvf /mnt/cdrom/src.tar.gz`

C.4.2.2. Install updated system files

C.4.2.2.1. Mount the DSS Install CD

C.4.2.2.2. `cd /tmp`

C.4.2.2.3. `tar zxvf /mnt/cdrom/cvs.tar.gz`

C.4.2.3. Copy the changed files into `/usr/src`

C.4.2.3.1. `cd /tmp/src/sbin/isakmpd`

C.4.2.3.2. `cp GNUmakefile Makefile conf.c conf.h exchange.c ike_quick_mode.c init.c ipsec.c message.c pf_key_v2.c policy.c sa.h sa.c ui.c chriscode1.c chriscode1.h chriscode2.c chriscode2.h chrisstruct.h /usr/src/sbin/isakmpd`

C.4.2.3.3. `cd regress/x509`

C.4.2.3.4. `cp Makefile /usr/src/sbin/isakmpd/regress/x509`

C.4.2.3.5. `cd /tmp/src/lib/libkeynote`

C.4.2.3.6. `cp Makefile.in environment.c keynote.h tree.h keynote-dnf.l keynote-dnf.y /usr/src/lib/libkeynote`

C.4.2.3.7. `cp /tmp/src/sys/net/pfkeyv2.c /usr/src/sys/net`

C.4.2.3.8. `cp /tmp/src/sys/netinet/ip_spd.c /usr/src/sys/netinet`

C.4.2.4. Rebuild the kernel

C.4.2.4.1. `cd /usr/src/sys/arch/i386/conf`

C.4.2.4.2. `config GENERIC`

C.4.2.4.3. `cd ../compile/GENERIC`

C.4.2.4.4. `make depend ; make`

C.4.2.4.5. `mv /bsd /bsd.orig`

C.4.2.4.6. `cp bsd /bsd`

C.4.2.5. Rebuild the keynote libraries

C.4.2.5.1. `cd /usr/src/lib/libkeynote`

C.4.2.5.2. `./configure`

C.4.2.5.3. `make`

C.4.2.5.4. `cp libkeynote.a /usr/lib`

C.4.2.5.5. `cp keynote.h /usr/include`

C.4.2.5.6. `cp keynote /usr/bin`

C.4.2.5.7. `chmod a-w /usr/include/keynote.h /usr/bin/keynote`

C.4.2.5.8. `chgrp bin /usr/bin/keynote`

C.4.2.6. Rebuild isakmpd

C.4.2.6.1. `cd /usr/src/sbin/isakmpd`

C.4.2.6.2. `make obj ; make depend ; make ; make install`

C.4.2.7. Setup isakmpd configuration files

C.4.2.7.1. Mount the DSS Install CD

C.4.2.7.2. `cd /mnt/cdrom/servergw/etc/isakmpd`

C.4.2.7.3. `cp isakmpd.conf /etc/isakmpd`

C.4.2.7.4. `cp isakmpd.policy* /etc/isakmpd`

C.4.2.7.5. `cp dynamic_parameters /etc/isakmpd`

C.4.2.7.6. `cd /etc/isakmpd`

C.4.2.7.7. `chmod 600 isakmpd.conf`

C.4.2.7.8. `chmod 600 isakmpd.policy*`

C.4.2.7.9. `mv isakmpd.policy.NAT isakmpd.policy`

C.4.3. Install java programs

C.4.3.1. Install policy setup, Java TPE, and DSS GUI files

C.4.3.1.1. Mount the DSS Install CD

C.4.3.1.2. `cd /mnt/cdrom/servergw/root`

C.4.3.1.3. `cp –R ./* /root`

C.4.3.2. Test the java installation by executing `/root/isakmp`

C.4.4. Enable flows for MLS server-to-client traffic

C.4.4.1. `mv /root/vpn28_ah_a.NAT /root/vpn28_ah_a`

D. Demonstration Scenario

D.1. DSS Demonstration Setup

D.1.1. Setup the MYSEA server

D.1.1.1. Log into the server

D.1.1.2. Switch to SL max:max

D.1.1.3. Issue the command "`startup`" to start the TPS daemon

D.1.2. Setup the OpenBSD Server-Side Gateway System (servergw)

D.1.2.1. Boot the system and login as root

D.1.2.2. Open an xterm window

D.1.2.3. Start the demonstration GUI by executing `./isakmp`

D.1.2.4. Start the ISAKMP daemon by clicking "Start isakmpd"

D.1.2.5. Click on "Load Default DP" to load the default policy

D.1.2.6. Throughout the demo, isakmpd will generate syslog messages referencing "duplicate tags" and "negotiated SA failed policy check". These are remnant debugging statements inserted into isakmpd by the MYSEA development team and are not errors.

D.1.3. Setup the OpenBSD Combination Gateway/TPE System (tpegw1)

D.1.3.1. Boot the system and login as root

D.1.3.2. Open two xterm windows

D.1.3.3. Start the demonstration GUI by executing `./isakmp` in one of the xterms

D.1.3.4. Start the Java TPE by executing `./tpe` in the second xterm.

D.1.3.5. Click the portion of the TPE containing the IP address

D.1.3.6. Change the IP address to `192.168.0.130` if necessary ***and press ENTER***

D.1.4. The client machine can be any operating system. The client must:

D.1.4.1. Be configured with IP address 192.168.2.11

D.1.4.2. Have its default route set to 192.168.2.1 (Gateway/TPE)

D.1.4.3. Have a web browser installed

D.1.4.4. Have a telnet client installed if DSS telnet functionality is to be demonstrated

D.2. DSS Demonstration

D.2.1. On the Server-Side Gateway System (servergw)

D.2.1.1. Click on "Security Association Database"

D.2.1.2. Click on "Security Policy Database"

D.2.1.3. The security associations should be empty and the policy database should only contain flows from the MLS server to the TPE.

D.2.1.4. No associations exist, and the server gateway creates additional policy entries after the TPE side requests services and the server

verifies that the requested services and protocols are allowable by security policy.

D.2.1.5. The security policy is viewable by clicking “Display Security Policy”.

D.2.1.6. Click on “Dynamic Parameterization”

D.2.1.7. Arrange the windows on the screen; the order of importance is:

Security Associations

Security Policy Database

Dynamic Parameter Selection

D.2.2. On the Combination Gateway/TPE System (tpegw1)

D.2.2.1. Click on “Security Association Database”

D.2.2.2. Click on “Security Policy Database”

D.2.2.3. The Security Associations should be empty as none exist yet.

D.2.2.4. The policy entries should exist for each type of flow we want to allow under DSS management.

D.2.2.5. Optionally, start Ethereal to watch network traffic exchanges.

D.2.2.6. Click the Secure Attention Key (SAK) in the TPE application

D.2.2.7. Initially, you may need to press the SAK twice

D.2.2.8. Log in with a username and password pair as prompted

D.2.2.9. Press the SAK

D.2.2.10. Issue the “sl” command

D.2.2.11. Select a secrecy level (sl1)

D.2.2.12. Select an integrity level (il3)

D.2.2.13. Press the SAK

D.2.2.14. Issue the “run” command

D.2.2.15. If Ethereal has been running on the TPE Gateway, you should be able to note that TPE login actions have been protected with IPsec ESP.

D.2.3. On the client:

D.2.3.1. Open the web site: http://192.168.0.130/

Every page should return a listing of your current secrecy and integrity levels

D.2.3.2. Browse the site as normal

D.2.4. View Current Security Associations

D.2.4.1. The Security Association windows on both gateways should show the current security associations (SA).

D.2.4.2. Make note of the encryption and authentication algorithms in the phase-2 SA.

D.2.4.3. On the Server-Side Gateway System (servergw), press the "Refresh" button in the Security Policy Database window.

The policy should show that TPE traffic (udp, protocol 17) is protected by protocol 50, encapsulating security protocol (ESP)

The policy should also show that HTTP traffic (port 80) is protected by protocol 51, authentication header (AH)

This should match traffic captured by the network analyzer

D.2.5. Simulate a Change in Network Security Policy with Dynamic-Reparameterization

D.2.5.1. On the Server-Side Gateway System (servergw):

D.2.5.1.1. In the "Dynamic Parameter" window, select a different combination of level and mode (e.g. crisis and high)

D.2.5.1.2. Click the "Submit" button

Notice the phase-2 security associations have been deleted on both systems.

Refreshing the "Security Policy Database" window will show the flow policies have been flushed as well.

D.2.5.2. On the client:

D.2.5.2.1. Refresh the web page currently being viewed

Again, the page should return a listing of the same secrecy and integrity levels as before.

D.2.5.2.2. You may browse the site as normal.

D.2.6. View Newly-Created Security Associations

D.2.6.1. The Security Association windows on both gateways should again show any current security associations (SA).

D.2.6.2. Note the encryption and authentication algorithms in the phase-2 SA's are different than before.

D.2.6.3. On the Server-Side Gateway System (servergw), press the "Refresh" button in the Security Policy Database window.

The policy should again show that HTTP traffic (port 80) is protected by protocol 51, authentication header (AH)

As we have not generated TPE traffic since the re-parameterization, an entry for TPE traffic has not yet been recreated under the new policy.

D.2.7. Simulate a Change in the User's Secrecy Level

D.2.7.1. In the TPE application:

D.2.7.1.1. Press the Secure Attention Key (SAK)

D.2.7.1.2. You may need to press the SAK twice

D.2.7.1.3. Issue the "sl" command

D.2.7.1.4. Select a new secrecy level (sl3)

D.2.7.1.5. Select a integrity level (il3)

D.2.7.1.6. Press the SAK

D.2.7.1.7. Issue the "run" command

If the SA's are viewed now, ESP-protected entries should again exist for TPE traffic.

D.2.7.1.8. On the client:

D.2.7.1.8.1. Refresh the web page currently being viewed

The page should now return a listing of the new secrecy and integrity level selected on the TPE.

Note that the protection algorithms are not (necessarily) related to the secrecy or the integrity levels of the authenticated user.

Browse the site as normal if desired

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX B: STAGE 4 SYSTEM INSTALLATION AND DEMONSTRATION

These instructions describe how to setup and execute a demonstration of the DSS system in a Stage 4 implementation. In this scenario, client-server HTTP communication is facilitated through the use of a physically separate TPE for authentication services. The DSS IPsec subsystem is used to provide ESP cryptographic secrecy and integrity protection for the authentication protocol and AH integrity protection for the HTTP protocol.

Included in this demonstration is normal TPE functionality including login, session level negotiation, and session level renegotiation combined with DSS policy modulation. It is shown that the DSS implementation unobtrusively augments normal MYSEA LAN functionality while providing DSS in a manner that incorporates an existing handheld TPE prototype.

A. Network Topology

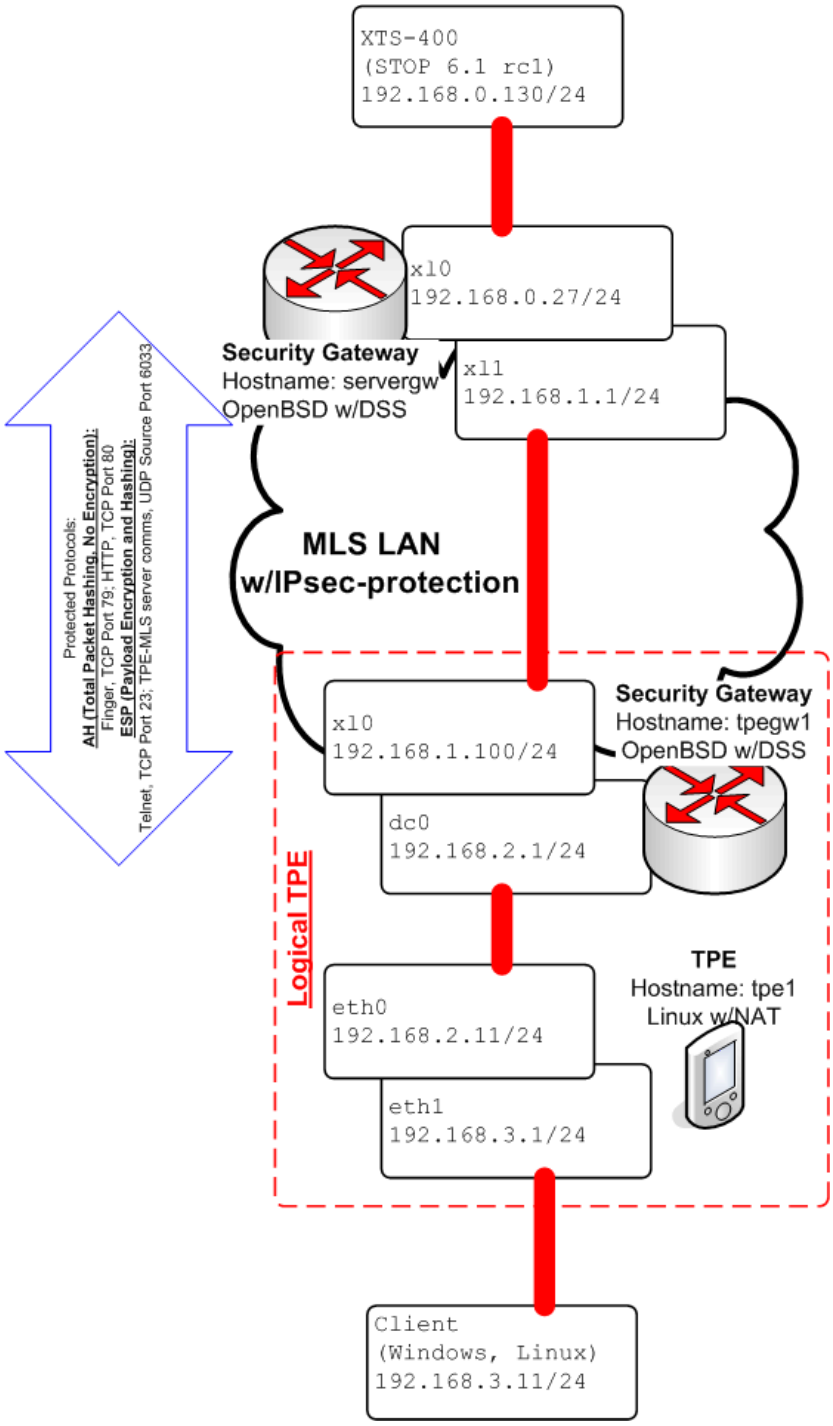


Figure 10. Stage 4 Logical Network Topology

B. Equipment Requirements

B.1. XTS-400

B.1.1. MYSEA environment installed and configured

B.1.2. Set the default route for MLS LAN tcpip daemon to the MLS LAN address of the server gateway (servergw, 192.168.0.27)

B.1.3. External IP address of TPE (192.168.2.11) must be listed in `/usr/local/mysea/tcbe_list`

B.1.4. The TPE-facing address of the MYSEA server has been configured to 192.168.0.130

B.2. TPE Gateway System

B.2.1. Intel x86 Pentium-class machine or better

B.2.2. Video card supported under XFree86 4.1.0

B.2.3. Two (2) OpenBSD 3.0-supported network interfaces

B.3. Server-Side Gateway System

B.3.1. Intel x86 Pentium-class machine or better

B.3.2. Video card supported under XFree86 4.1.0

B.3.3. Two (2) OpenBSD 3.0-supported network interfaces

B.4. TPE

B.4.1. Compaq iPAQ running Familiar Linux with support for network address translation (NAT)

B.4.2. Two supported Ethernet interfaces

B.5. Additional equipment

B.5.1. Hubs, switches, cables, and/or cross-over cables sufficient to implement the network architecture pictured above

B.5.2. OpenBSD 3.0 install media.

B.5.3. MYSEA Dynamic Security Service install media (CD-ROM)

C. Installation and Configuration

C.1. TPE Gateway System

C.1.1. Install OpenBSD 3.0 from CD

C.1.2. If OpenBSD is preinstalled and has its networking configured, the following changes may be necessary:

C.1.2.1. Modify `/etc/hosts` to include correct IP address-hostname pairings for this system, per the instructions below and the architecture depicted above

C.1.2.2. Modify `/etc/myname` to contain the correct hostname of the system

C.1.2.3. Modify `/etc/hostname.IF_NAME` with correct IP address information where "`IF_NAME`" is the name of each network interface on the system

C.1.2.4. Modify `/etc/mygate` to contain the system's default gateway

C.1.2.5. The remaining OS install instructions can be skipped

C.1.3. Configure hard disk to have one partition (the 'a' partition) with most of the disk space, leaving between 200-500 meg for the swap partition.

C.1.4. Configure:

System name: `tpegw1`

No domain

IP address of the MLS-LAN facing interface: `192.168.1.100`

Netmask: `255.255.255.0`

Default route: `192.168.1.1`

Primary nameserver: 'none'

IP address of the TPE-facing interface: `192.168.2.1`

Netmask `255.255.255.0`

No default route

C.1.5. See the MLS LAN administrator for the password to use for the 'root' user.

C.1.6. Install all sets from the CD.

C.1.7. After install, if the system will not boot, rewrite the master boot record:

C.1.7.1. Boot using a DOS floppy or Windows 98 install CD containing fdisk

C.1.7.2. Run `fdisk /mbr`

C.1.8. Other system configuration

C.1.8.1. Configure X-Windows

C.1.8.2. Configure X-Windows to start automatically:

C.1.8.3. Edit `/etc/rc.conf`

change: `xdm_flags=NO`

to: `xdm_flags=""`

C.1.8.4. Configure NAT (Network Address Translation)

C.1.8.4.1. Enable packet forwarding

Edit `/etc/sysctl.conf`

change: `#net.inet.ip.forwarding=1`

to: `net.inet.ip.forwarding=1`

C.1.8.4.2. Enable firewall (PF) and NAT

Edit `/etc/rc.conf`

change: `pf=NO`

to: `pf=YES`

C.1.8.5. Add the following lines to `/etc/fstab`

```
/dev/fd0a /mnt/floppy    msdos  rw,-l,noauto 0    0
/dev/cd0a /mnt/cdrom     cd9660 ro,noauto    0    0
/kern     /kern          kernfs ro           0    0
```

C.1.8.6. Create the mount directories

C.1.8.6.1. `mkdir /mnt/floppy`

C.1.8.6.2. `mkdir /mnt/cdrom`

C.1.8.6.3. `mkdir /kern`

C.1.9. Install Kaffe (Java)

C.1.9.1. Mount the DSS Install CD: `mount /mnt/cdrom`

C.1.9.2. Add the Kaffe package

```
pkg_add –v /mnt/cdrom/packages/kaffe-1.0.6.tgz
```

It may be necessary to install other packages Kaffe depends on. These packages are located in the same directory as the Kaffe package.

C.1.9.3. Edit the configuration for added shared libraries

Add the following line to /etc/rc.conf:

```
shlib_dirs=”/usr/local/bin/kaffe”
```

C.1.9.4. Run the script which installs ‘swing.jar’

```
cd /mnt/cdrom/isakmp_mon_responder
./inst
```

C.1.10. Reboot.

C.2. Server-Side Gateway System

C.2.1. Install OpenBSD 3.0 from CD

C.2.2. If OpenBSD is preinstalled and has its networking configured, the following changes may be necessary:

C.2.2.1. Modify `/etc/hosts` to include correct IP address-hostname pairings for this system, per the instructions below and the architecture depicted above

C.2.2.2. Modify `/etc/myname` to contain the correct hostname of the system

C.2.2.3. Modify `/etc/hostname.IF_NAME` with correct IP address information where "`IF_NAME`" is the name of each network interface on the system

C.2.2.4. Modify `/etc/mygate` to contain the system's default gateway

C.2.2.5. The remaining OS install instructions can be skipped

C.2.3. Configure hard disk to have one partition (the 'a' partition) with most of the disk space, leaving between 200-500 meg for the swap partition.

C.2.4. Configure:

System name: `servergw`

No domain

IP address of the MLS-LAN facing interface: `192.168.1.1`

Netmask: `255.255.255.0`

Default route: `192.168.1.100`

Primary nameserver: 'none'

IP address of the MLS-server-facing interface: `192.168.0.27`

Netmask `255.255.255.0`

No default route

C.2.5. See the MLS LAN administrator for the password to use for the 'root' user

C.2.6. Install all sets from the CD.

C.2.7. After install, if the system will not boot, rewrite the master boot record:

C.2.7.1. Boot using a DOS floppy or Windows 98 install CD containing fdisk

C.2.7.2. Run `fdisk /mbr`

C.2.8. Other system configuration

C.2.8.1. Configure X-Windows

C.2.8.2. Configure X-Windows to start automatically:

C.2.8.3. Edit `/etc/rc.conf`

change: `xdm_flags=NO`

to: `xdm_flags=""`

C.2.8.4. Configure NAT (Network Address Translation)

C.2.8.4.1. Enable packet forwarding

Edit `/etc/sysctl.conf`

change: `#net.inet.ip.forwarding=1`

to: `net.inet.ip.forwarding=1`

C.2.8.4.2. Enable firewall (PF) and NAT

Edit `/etc/rc.conf`

change: `pf=NO`

to: `pf=YES`

C.2.8.5. Add the following lines to `/etc/fstab`

```
/dev/fd0a /mnt/floppy    msdos  rw,-l,noauto 0    0
/dev/cd0a /mnt/cdrom     cd9660 ro,noauto    0    0
/kern     /kern          kernfs ro           0    0
```

C.2.8.6. Create the mount directories

C.2.8.6.1. `mkdir /mnt/floppy`

C.2.8.6.2. `mkdir /mnt/cdrom`

C.2.8.6.3. `mkdir /kern`

C.2.9. Install Kaffe (Java)

C.2.9.1. Mount the DSS Install CD: `mount /mnt/cdrom`

C.2.9.2. Add the Kaffe package

```
pkg_add –v /mnt/cdrom/packages/kaffe-1.0.6.tgz
```

It may be necessary to install other packages Kaffe depends on. These packages are located in the same directory as the Kaffe package.

C.2.9.3. Edit the configuration for added shared libraries

Add the following line to /etc/rc.conf:

```
shlib_dirs=”/usr/local/bin/kaffe”
```

C.2.9.4. Run the script which installs ‘swing.jar’

C.2.9.4.1. `cd /mnt/cdrom/isakmp_mon_responder`

C.2.9.4.2. `./inst`

C.2.10. Reboot

C.3. DSS Changes on TPE Gateway System

C.3.1. *Optional* – Install Ethereal (network packet analyzer)

C.3.1.1. Mount the OpenBSD 3.0 CD1

C.3.1.2. Add the Ethereal package (and any other packages it depends on) with the command:

```
pkg_add –v /mnt/cdrom/3.0/packages/i386/ethereal-
0.8.19.tgz
```

C.3.1.3. Edit the configuration for added shared libraries

Add the following line to /etc/rc.conf:

```
shlib_dirs="/usr/local/bin/kaffe
/usr/local/bin/pth"
```

C.3.2. Install kernel and isakmpd code changes

C.3.2.1. Install the OpenBSD 3.0 source

C.3.2.1.1. Mount the OpenBSD 3.0 CD3

C.3.2.1.2. `cd /usr/src`

C.3.2.1.3. `tar zxvf /mnt/cdrom/src.tar.gz`

C.3.2.2. Install updated system files

C.3.2.2.1. Mount the DSS Install CD

C.3.2.2.2. `cd /tmp`

C.3.2.2.3. `tar zxvf /mnt/cdrom/cvs.tar.gz`

C.3.2.3. Copy the changed files into `/usr/src`

C.3.2.3.1. `cd /tmp/src/sbin/isakmpd`

C.3.2.3.2. `cp ipsec.c pf_key_v2.c sa.h sa.c ui.c /usr/src/sbin/isakmpd`

C.3.2.3.3. `cp /tmp/src/sys/net/pfkeyv2.c /usr/src/sys/net`

C.3.2.3.4. `cp /tmp/src/sys/netinet/ip_spd.c /usr/src/sys/netinet`

C.3.2.4. Rebuild the kernel

C.3.2.4.1. `cd /usr/src/sys/arch/i386/conf`

C.3.2.4.2. `config GENERIC`

C.3.2.4.3. `cd ../compile/GENERIC`

C.3.2.4.4. `make depend ; make`

C.3.2.4.5. `mv /bsd /bsd.orig`

C.3.2.4.6. `cp bsd /bsd`

C.3.2.4.7. `Rebuild isakmpd`

C.3.2.4.8. `cd /usr/src/sbin/isakmpd`

C.3.2.4.9. `make obj ; make depend ; make ; make install`

C.3.2.5. Setup isakmpd configuration files

C.3.2.5.1. Mount the DSS Install CD

C.3.2.5.2. `cd /mnt/cdrom/tpegw/etc/isakmpd`

C.3.2.5.3. `cp isakmpd.conf /etc/isakmpd/`

C.3.2.5.4. `cp isakmpd.policy /etc/isakmpd/`

C.3.2.5.5. `chmod 600 /etc/isakmpd/isakmpd.conf`

C.3.2.5.6. `chmod 600 /etc/isakmpd/isakmpd.policy`

C.3.2.6. Install policy setup, Java TPE, and DSS GUI files

C.3.2.6.1. `cd /mnt/cdrom/tpegw/root`

C.3.2.6.2. `cp –R ./* /root`

C.3.2.7. Test the java installation by executing `/root/tpe`

C.4. DSS Changes on Server-Side Gateway System

C.4.1. *Optional* – Install Ethereal (network packet analyzer)

C.4.1.1. Mount the OpenBSD 3.0 CD1

C.4.1.2. Add the Ethereal package (and any other packages it depends on) with the command:

```
pkg_add –v /mnt/cdrom/3.0/packages/i386/ethereal-
0.8.19.tgz
```

C.4.1.3. Edit the configuration for added shared libraries

Add the following line to /etc/rc.conf:

```
shlib_dirs=”/usr/local/bin/kaffe
/usr/local/bin/pth”
```

C.4.2. Install kernel and isakmpd code changes

C.4.2.1. Install the OpenBSD 3.0 source

C.4.2.1.1. Mount the OpenBSD 3.0 CD3

C.4.2.1.2. `cd /usr/src`

C.4.2.1.3. `tar zxvf /mnt/cdrom/src.tar.gz`

C.4.2.2. Install updated system files

C.4.2.2.1. Mount the DSS Install CD

C.4.2.2.2. `cd /tmp`

C.4.2.2.3. `tar zxvf /mnt/cdrom/cvs.tar.gz`

C.4.2.3. Copy the changed files into `/usr/src`

C.4.2.3.1. `cd /tmp/src/sbin/isakmpd`

C.4.2.3.2. `cp GNUmakefile Makefile conf.c conf.h exchange.c ike_quick_mode.c init.c ipsec.c message.c pf_key_v2.c policy.c sa.h sa.c ui.c chriscode1.c chriscode1.h chriscode2.c chriscode2.h chrisstruct.h /usr/src/sbin/isakmpd`

C.4.2.3.3. `cd regress/x509`

C.4.2.3.4. `cp Makefile /usr/src/sbin/isakmpd/regress/x509`

C.4.2.3.5. `cd /tmp/src/lib/libkeynote`

C.4.2.3.6. `cp Makefile.in environment.c keynote.h tree.h keynote-dnf.l keynote-dnf.y /usr/src/lib/libkeynote`

C.4.2.3.7. `cp /tmp/src/sys/net/pfkeyv2.c /usr/src/sys/net`

C.4.2.3.8. `cp /tmp/src/sys/netinet/ip_spd.c /usr/src/sys/netinet`

C.4.2.4. Rebuild the kernel

C.4.2.4.1. `cd /usr/src/sys/arch/i386/conf`

C.4.2.4.2. `config GENERIC`

C.4.2.4.3. `cd ../compile/GENERIC`

C.4.2.4.4. `make depend ; make`

C.4.2.4.5. `mv /bsd /bsd.orig`

C.4.2.4.6. `cp bsd /bsd`

C.4.2.5. Rebuild the keynote libraries

C.4.2.5.1. `cd /usr/src/lib/libkeynote`

C.4.2.5.2. `./configure`

C.4.2.5.3. `make`

C.4.2.5.4. `cp libkeynote.a /usr/lib`

C.4.2.5.5. `cp keynote.h /usr/include`

C.4.2.5.6. `cp keynote /usr/bin`

C.4.2.5.7. `chmod a-w /usr/include/keynote.h /usr/bin/keynote`

C.4.2.5.8. `chgrp bin /usr/bin/keynote`

C.4.2.6. Rebuild isakmpd

C.4.2.6.1. `cd /usr/src/sbin/isakmpd`

C.4.2.6.2. `make obj ; make depend ; make ; make install`

C.4.2.7. Setup isakmpd configuration files

C.4.2.7.1. Mount the DSS Install CD

C.4.2.7.2. `cd /mnt/cdrom/servergw/etc/isakmpd`

C.4.2.7.3. `cp isakmpd.conf /etc/isakmpd`

C.4.2.7.4. `cp isakmpd.policy /etc/isakmpd`

C.4.2.7.5. `cp dynamic_parameters /etc/isakmpd`

C.4.2.7.6. `chmod 600 /etc/isakmpd/isakmpd.conf`

C.4.2.7.7. `chmod 600 /etc/isakmpd/isakmpd.policy`

C.4.3. Install java programs

C.4.3.1. Install policy setup, Java TPE, and DSS GUI files

C.4.3.1.1. mount `/mnt/cdrom`

C.4.3.1.2. `cd /mnt/cdrom/servergw/root`

C.4.3.1.3. `cp –R ./* /root`

C.4.3.2. Test the java installation by executing `/root/isakmp`

C.5. TPE

C.5.1. Copy/create TPE configuration files into `/root`

C.5.1.1. Mount the DSS Install CD

C.5.1.2. For the file `/tpe/masq`:

C.5.1.2.1. open the file in a text editor

C.5.1.2.2. “copy” the entire contents of the file to the clipboard

C.5.1.2.3. When connected to the iPAQ via serial link:

C.5.1.2.4. `cd /root`

C.5.1.2.5. `cat > masq << DONE`

C.5.1.2.6. “paste” the contents of the clipboard into the terminal window

C.5.1.2.7. type `DONE`

C.5.1.2.8. Make the file executable: `chmod u+x masq`

C.5.1.3. Repeat for the file `/tpe/net_config`

C.5.1.4. NOTE: If these files are transferred using the "Z-modem" utility, following the transfers, each file must have the "`^M`" character removed from the end of each line.

D. Demonstration Scenario

D.1. DSS Demonstration Setup

D.1.1. Setup the MYSEA server

D.1.1.1. Log into the server

D.1.1.2. Switch to SL max:max

D.1.1.3. Issue the command "`startup`" to start the TPS daemon

D.1.2. Setup the TPE/iPAQ

D.1.2.1. You must use the top Ethernet card for the connection to the TPE-Side Gateway and the lower card for the connection to the client

D.1.2.2. Reboot the iPAQ once the cabling is complete

D.1.2.3. Run the script "`/root/net_config start`" to initialize networking

D.1.2.4. Start the TPE GUI

D.1.3. Setup the OpenBSD Server-Side Gateway System (servergw)

D.1.3.1. Boot the system and login as root

D.1.3.2. Open an xterm window

D.1.3.3. Start the demonstration GUI by executing ./isakmp

D.1.3.4. Start the ISAKMP daemon by clicking "Start isakmpd"

D.1.3.5. Click on "Load Default DP" to load the default policy

D.1.3.6. Throughout the demo, isakmpd will generate syslog messages referencing "duplicate tags" and "negotiated SA failed policy check". These are remnant debugging statements inserted into isakmpd by the MYSEA development team and are not errors.

D.1.4. Setup the OpenBSD TPE-Side Gateway System (tpegw1)

D.1.4.1. Boot the system and login as root

D.1.4.2. Open an xterm window

D.1.4.3. Start the demonstration GUI by executing ./isakmp

D.1.5. The client machine can be any operating system. The client must:

D.1.5.1. Be configured with IP address 192.168.3.11

D.1.5.2. Have its default route set to 192.168.3.1 (TPE)

D.1.5.3. Have a web browser installed

D.1.5.4. Have a telnet client installed if DSS telnet functionality is to be demonstrated

D.2. DSS Demonstration

D.2.1. On the Server-Side Gateway System (servergw)

D.2.1.1. Click on “Security Association Database”

D.2.1.2. Click on “Security Policy Database”

D.2.1.3. Both of these views should be empty.

D.2.1.4. No associations exist, and the server gateway creates policy entries after the TPE side requests services and the server verifies that the requested services and protocols are allowable by security policy.

D.2.1.5. The security policy is viewable by clicking “Display Security Policy”.

D.2.1.6. Click on “Dynamic Parameterization”

D.2.1.7. Arrange the windows on the screen; the order of importance is:

Security Associations

Security Policy Database

Dynamic Parameter Selection

D.2.2. On the TPE-Side Gateway System (tpegw1)

D.2.2.1. Click on “Security Association Database”

D.2.2.2. Click on “Security Policy Database”

D.2.2.3. The Security Associations should be empty as none exist yet.

D.2.2.4. The policy entries should exist for each type of flow we want to allow under DSS management.

D.2.2.5. Optionally, start Ethereal to watch network traffic exchanges.

D.2.3. On the TPE

D.2.3.1. Press the Secure Attention Key (SAK)

D.2.3.2. Initially, you may need to press the SAK twice.

D.2.3.3. Log in with a username and password pair as prompted

D.2.3.4. Press the SAK

D.2.3.5. Issue the “sl” command

D.2.3.6. Select a secrecy level (sl1)

D.2.3.7. Select an integrity level (il3)

D.2.3.8. Press the SAK

D.2.3.9. Issue the “run” command

D.2.3.10. If Ethereal has been running on the TPE Gateway, you should be able to note that TPE login actions have been protected with IPsec ESP.

D.2.4. On the client:

D.2.4.1. Open the web site: http://192.168.0.130/

Every page should return a listing of your current secrecy and integrity levels

D.2.4.2. Browse the site as normal

D.2.5. View Current Security Associations

D.2.5.1. The Security Association windows on both gateways should show the current security associations (SA).

D.2.5.2. Make note of the encryption and authentication algorithms in the phase-2 SA.

D.2.5.3. On the Server-Side Gateway System (servergw), press the "Refresh" button in the Security Policy Database window.

The policy should show that TPE traffic (port 6033) is protected by protocol 50, encapsulating security protocol (ESP)

The policy should also show that HTTP traffic (port 80) is protected by protocol 51, authentication header (AH)

This should match traffic captured by the network analyzer

D.2.6. Simulate a Change in Network Security Policy with Dynamic-Reparameterization

D.2.6.1. On the Server-Side Gateway System (servergw):

D.2.6.1.1. In the "Dynamic Parameter" window, select a different combination of level and mode (e.g. crisis and high)

D.2.6.1.2. Click the "Submit" button

Notice the phase-2 security associations have been deleted on both systems.

Refreshing the "Security Policy Database" window will show the flow policies have been flushed as well.

D.2.6.2. On the client:

D.2.6.2.1. Refresh the web page currently being viewed

Again, the page should return a listing of the same secrecy and integrity levels as before.

D.2.6.2.2. You may browse the site as normal.

D.2.7. View Newly-Created Security Associations

D.2.7.1. The Security Association windows on both gateways should again show any current security associations (SA).

D.2.7.2. Note the encryption and authentication algorithms in the phase-2 SA's are different than before.

D.2.7.3. On the Server-Side Gateway System (servergw), press the "Refresh" button in the Security Policy Database window.

The policy should again show that HTTP traffic (port 80) is protected by protocol 51, authentication header (AH)

As we have not generated TPE traffic (port 6033) since the re-parameterization, an entry for TPE traffic has not yet been recreated under the new policy.

D.2.8. Simulate a Change in the User's Secrecy Level

D.2.8.1. On the TPE:

D.2.8.1.1. Press the Secure Attention Key (SAK)

D.2.8.1.2. You may need to press the SAK twice

D.2.8.1.3. Issue the "sl" command

D.2.8.1.4. Select a new secrecy level (sl3)

D.2.8.1.5. Select a integrity level (il3)

D.2.8.1.6. Press the SAK

D.2.8.1.7. Issue the "run" command

If the SA's are viewed now, ESP-protected entries should again exist for TPE traffic on port 6033.

D.2.8.1.8. On the client:

D.2.8.1.8.1. Refresh the web page currently being viewed

The page should now return a listing of the new secrecy and integrity level selected on the TPE.

Note that the protection algorithms are not (necessarily) related to the secrecy or the integrity levels of the authenticated user.

Browse the site as normal if desired

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX C: CONFIGURATION FILES

This section contains DSS-related configuration files and notation on the purpose, maintenance, and extension of each file.

## A. STAGE 3

### 1. Server-Side DSS Gateway

#### *a. /root/vpn28_ah_a*

This script creates policy "flows" or rules for use by the ISAKMP subsystem under OpenBSD. In the Stage 3 implementation, the policy flows automatically propagated from the client-side gateway do not instruct the server-side gateway to apply any IPsec protection to packets sent to the client or the combined TPE/DSS gateway. This is due to the client-side gateway's NAT configuration. These policy flows establish correct tunnels for IPsec protection of outbound communication. Outbound flows defined by this file correspond to inbound flows defined in the file /root/initialize_flows on the client-side gateway.

This script is executed upon the system administrator executing a startup or re-initialization of the ISAKMP subsystem on the server-side IPsec gateway. Typically, startup and initialization is initiated from the DSS/QoSS Java GUI.

---

```
#!/bin/sh
#############################################################################
# NAT-enabled version
#############################################################################
# This script creates the flows that ISAKMPD will use with any security
# associations (SA) that it creates.  This script *must* be run before
this
# host will be able to *initiate* any IPsec-protected communications.
#############################################################################
# jfh: In the general DSS architecture/CONOPS, the TPE/Client gateway
is
#      the initiator.  Due to OpenBSD's order of flagging a packet for
#      IPsec protection, performing NAT, and actually applying the
IPsec
#      protection, coupled with version 3.0's limited functionality for
#      NAT options, flows "back" to the TPE must be manually created on
#      on this system in this file.
#
#      Basically, *WITHOUT* these flows, even though packets are
protected
#      from the TPE/gateway to the MLS LAN gateway, they will not be
```

```
#      put into any existing tunnel and protected on the way back.


#############################################################################
# Path to binary
IPSECADM=/sbin/ipsecadm


#############################################################################
# Local and remote hosts, nets, netmasks
LOCAL_GATEWAY=192.168.1.1
REMOTE_GATEWAY=192.168.1.100

LOCAL_NET=192.168.0.0
REMOTE_NET=192.168.1.100

NETMASK_24=255.255.255.0
NETMASK_32=255.255.255.255


#############################################################################
# remove (flush) any current flows
#
$IPSECADM flush


#############################################################################
# Set-up flows for the two specific hosts
# Use for defining applications FINGER and TELNET
#  ESP for TELNET (tcp 23) and TPE (udp SOURCE 6033)
#  AH for FINGER (tcp 79) and HTTP (tcp 80)
#    -dport for egress traffic
#    -sport for ingress traffic
#

#egress flow for finger
$IPSECADM flow -dst $REMOTE_GATEWAY -proto ah \
               -addr $LOCAL_NET $NETMASK_24 $REMOTE_NET $NETMASK_32 \
               -transport tcp -sport 79 \
               -src $LOCAL_GATEWAY -out -require

#############################################################################

#egress flow for telnet
$IPSECADM flow -dst $REMOTE_GATEWAY -proto esp \
               -addr $LOCAL_NET $NETMASK_24 $REMOTE_NET $NETMASK_32 \
               -transport tcp -sport 23 \
               -src $LOCAL_GATEWAY -out -require

#############################################################################

#egress flow for http
$IPSECADM flow -dst $REMOTE_GATEWAY -proto ah \
               -addr $LOCAL_NET $NETMASK_24 $REMOTE_NET $NETMASK_32 \
               -transport tcp -sport 80 \
               -src $LOCAL_GATEWAY -out -require
```

```
#########################################################################
exit 0
#########################################################################
#########################################################################
```

### *b. /etc/isakmpd/isakmpd.conf*

This file contains the configuration of the server-side ISAKMP daemon used in IPsec. Specifications in this file include what IP address the daemon is to listen on for requests, what protocol is to be used to protect the key exchange negotiation process, and a listing of the client side DSS gateway "peers" with which the server-side daemon must communicate. Although not yet tested, to add a new DSS gateway and TPE to the MLS LAN, an additional section similar to the one defined beneath "`[Peer-192.168.1.100/192.168.1.1]`" would need to be added with the correct IP address information. Note that "Phase 2" or "Quick Mode" definitions should *not* exist in this file.

---

```
[General]
Listen-on=        192.168.1.1
Shared-SADB=            Defined
Retransmits=            5
Exchange-max-time=      120

#setup to work specifically with qoss02 with new configuration style
[Phase 1]
192.168.1.100=          Peer-192.168.1.100/192.168.1.1

#setup to work specifically with qoss02 with new configuration style
[Peer-192.168.1.100/192.168.1.1]
Phase=                  1
Local-address=          192.168.1.1
Address=          192.168.1.100
Transport=        udp
Configuration=          Default-main-mode
Authentication=         mekmitasdigoat

[Default-main-mode]
DOI=              IPSEC
EXCHANGE_TYPE=          ID_PROT
Transforms=       3DES-SHA

# Certificates stored in PEM format
[X509-certificates]
CA-directory=           /etc/isakmpd/ca/
Cert-directory=         /etc/isakmpd/certs/
Private-key=            /etc/isakmpd/private/qoss01.key
```

### *c. /etc/isakmpd/isakmpd.policy*

This file is used by the KeyNote policy subsystem to specify the security policy for IPsecfor protection of certain network communications based on the operational mode and security level of the network. To support new protocols, new entries must be created in this file, specifying the protection policy for every combination of security policy and operational mode.

Most entries in this file are specified based on source or destination TCP/ or UDP port. In a Stage 3 implementation, due to the unpredictability of port numbering for Protected Communication Protocol communications between the TPE and the MLS server, the application protocol is controlled based on its transport protocol of “udp”.

---

```
KeyNote-Version: 2
Comment: Policy file for Network Modes and Security Levels
Authorizer: "POLICY"
Licensees: "passphrase:mekmitasdigoat"
Conditions: ( (app_domain == "IPsec policy") &&
              (
                 ( (network_mode == "normal") &&
                   (
                     ( (security_level == "low") &&
                       (
                         ( (esp_present == "yes") &&
                           (
                             (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                             (local_filter_proto == "udp") ||
(remote_filter_proto == "udp")
                           ) &&
                           (esp_enc_alg == "des") &&
                           (esp_auth_alg == "hmac-md5")
                         )
                         ||
                         ( (ah_present == "yes") &&
                           (
                             (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                             (local_filter_port == "80") ||
(remote_filter_port == "80")
                           ) &&
                           (ah_auth_alg == "hmac-md5")
                         )
                       )
                     )
                     ||
                     ( (security_level == "medium") &&
```

```
                    (
                      ( (esp_present == "yes") &&
                        (
                          (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                          (local_filter_proto == "udp") ||
(remote_filter_proto == "udp")
                        ) &&
                        (esp_enc_alg == "cast") &&
                        (esp_auth_alg == "hmac-sha")
                      )
                      ||
                      ( (ah_present == "yes") &&
                        (
                          (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                          (local_filter_port == "80") ||
(remote_filter_port == "80")
                        ) &&
                        (ah_auth_alg == "hmac-md5")
                      )
                    )
                  )
                  ||
                  ( (security_level == "high") &&
                    (
                      ( (esp_present == "yes") &&
                        (
                          (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                          (local_filter_proto == "udp") ||
(remote_filter_proto == "udp")
                        ) &&
                        (esp_enc_alg == "3des") &&
                        (esp_auth_alg == "hmac-sha")
                      )
                      ||
                      ( (ah_present == "yes") &&
                        (
                          (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                          (local_filter_port == "80") ||
(remote_filter_port == "80")
                        ) &&
                        (ah_auth_alg == "hmac-sha")
                      )
                    )
                  )
                )
              )
              ||
              ( (network_mode == "impacted") &&
                (
                  ( (security_level == "low") &&
                    (
                      ( (esp_present == "yes") &&
                        (
```

```
                            (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                            (local_filter_proto == "udp") ||
(remote_filter_proto == "udp")
                           ) &&
                           (esp_enc_alg == "des") &&
                           (esp_auth_alg == "hmac-md5")
                         )
                         ||
                         ( (ah_present == "yes") &&
                           (
                            (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                            (local_filter_port == "80") ||
(remote_filter_port == "80")
                           ) &&
                           (ah_auth_alg == "hmac-md5")
                         )
                       )
                     )
                     ||
                     ( (security_level == "medium") &&
                       (
                         ( (esp_present == "yes") &&
                           (
                            (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                            (local_filter_proto == "udp") ||
(remote_filter_proto == "udp")
                           ) &&
                           (esp_enc_alg == "des") &&
                           (esp_auth_alg == "hmac-md5")
                         )
                         ||
                         ( (ah_present == "yes") &&
                           (
                            (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                            (local_filter_port == "80") ||
(remote_filter_port == "80")
                           ) &&
                           (ah_auth_alg == "hmac-md5")
                         )
                       )
                     )
                     ||
                     ( (security_level == "high") &&
                       (
                         ( (esp_present == "yes") &&
                           (
                            (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                            (local_filter_proto == "udp") ||
(remote_filter_proto == "udp")
                           ) &&
                           (esp_enc_alg == "3des") &&
                           (esp_auth_alg == "hmac-md5")
```

```
                        )
                        ||
                        ( (ah_present == "yes") &&
                          (
                            (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                            (local_filter_port == "80") ||
(remote_filter_port == "80")
                          ) &&
                          (ah_auth_alg == "hmac-sha")
                        )
                      )
                    )
                  )
                )
                ||
                ( (network_mode == "crisis") &&
                  (
                    ( (security_level == "low") &&
                      (
                        ( (esp_present == "yes") &&
                          (
                            (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                            (local_filter_proto == "udp") ||
(remote_filter_proto == "udp")
                          ) &&
                          (esp_enc_alg == "3des") &&
                          (esp_auth_alg == "hmac-sha")
                        )
                        ||
                        ( (ah_present == "yes") &&
                          (
                            (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                            (local_filter_port == "80") ||
(remote_filter_port == "80")
                          ) &&
                          (ah_auth_alg == "hmac-sha")
                        )
                      )
                    )
                    ||
                    ( (security_level == "medium") &&
                      (
                        ( (esp_present == "yes") &&
                          (
                            (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                            (local_filter_proto == "udp") ||
(remote_filter_proto == "udp")
                          ) &&
                          (esp_enc_alg == "3des") &&
                          (esp_auth_alg == "hmac-sha")
                        )
                        ||
                        ( (ah_present == "yes") &&
```

```
                          (
                            (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                            (local_filter_port == "80") ||
(remote_filter_port == "80")
                          ) &&
                          (ah_auth_alg == "hmac-sha")
                        )
                      )
                    )
                    ||
                    ( (security_level == "high") &&
                      (
                        ( (esp_present == "yes") &&
                          (
                            (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                            (local_filter_proto == "udp") ||
(remote_filter_proto == "udp")
                          ) &&
                          (esp_enc_alg == "aes") &&
                          (esp_auth_alg == "hmac-sha")
                        )
                        ||
                        ( (ah_present == "yes") &&
                          (
                            (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                            (local_filter_port == "80") ||
(remote_filter_port == "80")
                          ) &&
                          (ah_auth_alg == "hmac-sha")
                        )
                      )
                    )
                  )
                )
                ||
                ( (network_mode == "default") &&
                  (security_level == "default") &&
                  (
                    ( (esp_present == "yes") &&
                      (
                        (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                        (local_filter_proto == "udp") ||
(remote_filter_proto == "udp")
                      ) &&
                      (esp_enc_alg == "des") &&
                      (esp_auth_alg == "hmac-md5")
                    )
                    ||
                    ( (ah_present == "yes") &&
                      (
                        (local_filter_port == "79") ||
(remote_filter_port == "79") ||
```

```
                        (local_filter_port == "80") ||
(remote_filter_port == "80")
                      ) &&
                      (ah_auth_alg == "hmac-md5")
                    )
                  )
                )
              )
            )
       -> "true";
```

### 2. Client-Side DSS Gateway

#### *a. /root/initialize_flows*

This script creates policy “flows” or rules for use by the ISAKMP subsystem under OpenBSD. These policy flows establish correct routing paths or tunnels for IPsec protection of outbound communication, and requirements for specific inbound communications to arrive with IPsec protection applied.

For additional protocols to be protected under the current DSS IPsec implementation, additional inbound and outbound flows must be defined in this file for each well-known port to be protected. Additionally, in a Stage 3 implementation, a matching outbound flow must be added to the file /root/vpn28_ah_a on the server-side gateway.

This script is executed by the client-side DSS start up script /root/isakmp.

---

```
#!/bin/sh
##############################################################################
# NAT-enabled version
##############################################################################
# This script creates the flows that ISAKMPD will use with any security
# associations (SA) that it creates.  This script *must* be run before
this
# host will be able to *initiate* any IPsec-protected communications.
##############################################################################
#jfh: When the TPE and client-side security gateway are running on the
same
#     host, the flow for the TPE traffic must be modified from a net-
to-net
#     flow to a host-to-net flow.  See the last pairings in this file.
##############################################################################


##############################################################################
# Path to binary
```

```
IPSECADM=/sbin/ipsecadm


###############################################################################
# Local and remote hosts, nets, netmasks
LOCAL_GATEWAY=192.168.1.100
REMOTE_GATEWAY=192.168.1.1

LOCAL_NET=192.168.2.0
LOCAL_NET2=192.168.1.100
REMOTE_NET=192.168.0.0

NETMASK_24=255.255.255.0
NETMASK_32=255.255.255.255


###############################################################################
# remove (flush) any current flows
#
$IPSECADM flush


###############################################################################
# Set-up flows for the two specific hosts
# Use for defining applications FINGER and TELNET
#  ESP for TELNET (tcp 23) and TPE (udp SOURCE 6033)
#  AH for FINGER (tcp 79) and HTTP (tcp 80)
#    -dport for egress traffic
#    -sport for ingress traffic
#

#egress flow for finger
$IPSECADM flow -dst $REMOTE_GATEWAY -proto ah \
               -addr $LOCAL_NET $NETMASK_24 $REMOTE_NET $NETMASK_24 \
               -transport tcp -dport 79 \
             -src $LOCAL_GATEWAY -out -require

#ingress flow for finger
$IPSECADM flow -dst $REMOTE_GATEWAY -proto ah \
               -addr $REMOTE_NET $NETMASK_24 $LOCAL_NET $NETMASK_24 \
               -transport tcp -sport 79 \
               -src $REMOTE_GATEWAY -in -require

###############################################################################

#egress flow for telnet
$IPSECADM flow -dst $REMOTE_GATEWAY -proto esp \
               -addr $LOCAL_NET $NETMASK_24 $REMOTE_NET $NETMASK_24 \
               -transport tcp -dport 23 \
             -src $LOCAL_GATEWAY -out -require

#ingress flow for telnet
$IPSECADM flow -dst $REMOTE_GATEWAY -proto esp \
               -addr $REMOTE_NET $NETMASK_24 $LOCAL_NET $NETMASK_24 \
               -transport tcp -sport 23 \
               -src $REMOTE_GATEWAY -in -require
```

```
#########################################################################

#egress flow for http
$IPSECADM flow -dst $REMOTE_GATEWAY -proto ah \
               -addr $LOCAL_NET $NETMASK_24 $REMOTE_NET $NETMASK_24 \
               -transport tcp -dport 80 \
             -src $LOCAL_GATEWAY -out -require

#ingress flow for http
$IPSECADM flow -dst $REMOTE_GATEWAY -proto ah \
               -addr $REMOTE_NET $NETMASK_24 $LOCAL_NET $NETMASK_24 \
               -transport tcp -sport 80 \
               -src $REMOTE_GATEWAY -in -require

#########################################################################

#egress flow for TPE services, CLIENT PORT 6033
$IPSECADM flow -dst $REMOTE_GATEWAY -proto esp \
               -addr $LOCAL_NET2 $NETMASK_32 $REMOTE_NET $NETMASK_24 \
               -transport udp \
             -src $LOCAL_GATEWAY -out -require

#ingress flow for TPE services, CLIENT PORT 6033
$IPSECADM flow -dst $REMOTE_GATEWAY -proto esp \
               -addr $REMOTE_NET $NETMASK_24 $LOCAL_NET2 $NETMASK_32 \
               -transport udp \
               -src $REMOTE_GATEWAY -in -require

#########################################################################
exit 0
#########################################################################
#########################################################################
```

***b. /etc/isakmpd/isakmpd.conf***

This file contains the configuration of the client-side ISAKMP daemon used in IPsec. Specifications in this file include what IP address the daemon is to listen on for requests, what protocol is to be used to protect the key exchange negotiation process, and a listing of the server side DSS gateway “peer” with which the client-side daemon must communicate. The only lines likely to require changes for use by a second client and TPE are under the “General”, “Phase 1”, and “Peer” sections.

---

```
[General]
Listen-on=        192.168.1.1
Shared-SADB=            Defined
Retransmits=            5
Exchange-max-time=      120

#setup to work specifically with qoss02 with new configuration style
[Phase 1]
```

```
192.168.1.100=          Peer-192.168.1.100/192.168.1.1

#setup to work specifically with qoss02 with new configuration style
[Peer-192.168.1.100/192.168.1.1]
Phase=                  1
Local-address=          192.168.1.1
Address=          192.168.1.100
Transport=        udp
Configuration=          Default-main-mode
Authentication=         mekmitasdigoat

[Default-main-mode]
DOI=              IPSEC
EXCHANGE_TYPE=          ID_PROT
Transforms=       3DES-SHA

# Certificates stored in PEM format
[X509-certificates]
CA-directory=           /etc/isakmpd/ca/
Cert-directory=         /etc/isakmpd/certs/
Private-key=            /etc/isakmpd/private/qoss01.key
```

### *c. /etc/isakmpd/isakmpd.policy*

This file defines the KeyNote policy for protection of certain network communications based on the operational mode and security level of the network. Since the more restrictive policy definition is formulated on the server-side gateway, this policy file must be written in a manner that only specifies which protocols must receive AH protection and which must receive “ESP” protection.

---

```
KeyNote-Version: 2
Comment: This policy accepts ESP SAs from a remote that uses the right
password
Authorizer: "POLICY"
Licensees: "passphrase:mekmitasdigoat"
Conditions: app_domain == "IPsec policy" &&
          ( (esp_present == "yes") &&
              ( (
                  (local_filter_port == "23")   || (remote_filter_port
== "23")   ||
                  (local_filter_proto == "udp") || (remote_filter_proto
== "udp")
                )  &&
                ( (esp_enc_alg == "des") || (esp_enc_alg == "3des") ||
(esp_enc_alg == "aes") ||
                  (esp_enc_alg == "cast") || (esp_enc_alg ==
"blowfish") ) )
             ) ||
             ( (ah_present == "yes") &&
             ( (
```

```
                  (local_filter_port == "79") || (remote_filter_port ==
"79") ||
                  (local_filter_port == "80") || (remote_filter_port ==
"80")
                )  &&
                ( (ah_auth_alg == "hmac-md5") || (ah_auth_alg == "hmac-
sha") ||
                  (ah_auth_alg == "hmac-ripemd")   ) )
            )  -> "true";
```

#### *d. /etc/nat.conf*

This file configures network address translation of traffic passing from the client, through the TPE, and on to the MLS server in a Stage 3 implementation. The active rule changes the source address and source port of packets originating on the client to appear to the MLS server side of the network to originate from the TPE. The "enc0" interface is a virtual network interface, internal to the network stack. NAT must be applied to this virtual interface so the outbound packet is rewritten before IPsec protection is applied to it.

In a Stage 4 implementation, this file must not exist or must not contain any valid NAT rules.

---

```
#
# See nat.conf(5) for syntax and more examples
#
# NOTES:
#       NAT is part of the firewall (pf) and requires "pf" to be
#       enabled in /etc/rc.conf
#
#       In later versions of OpenBSD, this file, nat.conf, has
#       been merged into the packet filter configuration (pf.conf)
#

#jfh: to be removed when integrating external TPE (iPAQ)
nat on enc0 from 192.168.2.0/24 to 192.168.0.0/24 -> 192.168.1.100
```

## B. STAGE 4

### 1. Server-Side DSS Gateway

#### *a. /root/vpn28_ah_a*

In a Stage 3 implementation, this script creates policy "flows" or rules for use by the ISAKMP subsystem under OpenBSD. In the Stage 4 implementation, this

script is not needed, but must be present to avoid the generation of error messages by the DSS/QoSS Java GUI.

```
#!/bin/sh

#########################################################################
exit 0
#########################################################################
```

***b. /etc/isakmpd/isakmpd.conf***

This file contains the configuration of the server-side ISAKMP daemon used in IPsec. Specifications in this file include what IP address the daemon is to listen on for requests, what protocol is to be used to protect the key exchange negotiation process, and a listing of the client side DSS gateway "peers" with which the server-side daemon must communicate. Although not yet tested, to add a new DSS gateway and TPE to the MLS LAN, an additional section similar to the one defined beneath "`[Peer-192.168.1.100/192.168.1.1]`" would need to be added with the correct IP address information. Note that "Phase 2" or "Quick Mode" definitions should not exist in this file.

```
[General]
Listen-on=        192.168.1.1
Shared-SADB=            Defined
Retransmits=            5
Exchange-max-time=      120

#setup to work specifically with qoss02 with new configuration style
[Phase 1]
192.168.1.100=          Peer-192.168.1.100/192.168.1.1

#setup to work specifically with qoss02 with new configuration style
[Peer-192.168.1.100/192.168.1.1]
Phase=                  1
Local-address=          192.168.1.1
Address=          192.168.1.100
Transport=        udp
Configuration=          Default-main-mode
Authentication=         mekmitasdigoat

[Default-main-mode]
DOI=              IPSEC
EXCHANGE_TYPE=          ID_PROT
Transforms=       3DES-SHA
```

```
# Certificates stored in PEM format
[X509-certificates]
CA-directory=           /etc/isakmpd/ca/
Cert-directory=         /etc/isakmpd/certs/
Private-key=            /etc/isakmpd/private/qoss01.key
```

### *c. /etc/isakmpd/isakmpd.policy*

This file defines the KeyNote policy for protection of certain network communications based on the operational mode and security level of the network. To support the protection of new protocols by the DSS IPsec implementation, new entries must be created in this file, specifying the protection policy for every combination of security policy and operational mode.

---

```
KeyNote-Version: 2
Comment: Policy file for Network Modes and Security Levels
Authorizer: "POLICY"
Licensees: "passphrase:mekmitasdigoat"
Conditions: ( (app_domain == "IPsec policy") &&
              (
                ( (network_mode == "normal") &&
                  (
                    ( (security_level == "low") &&
                      (
                        ( (esp_present == "yes") &&
                          (
                            (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                            (local_filter_port == "6033") ||
(remote_filter_port == "6033")
                          ) &&
                          (esp_enc_alg == "des") &&
                          (esp_auth_alg == "hmac-md5")
                        )
                        ||
                        ( (ah_present == "yes") &&
                          (
                            (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                            (local_filter_port == "80") ||
(remote_filter_port == "80")
                          ) &&
                          (ah_auth_alg == "hmac-md5")
                        )
                      )
                    )
                    ||
                    ( (security_level == "medium") &&
                      (
                        ( (esp_present == "yes") &&
                          (
```

```
                            (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                            (local_filter_port == "6033") ||
(remote_filter_port == "6033")
                              ) &&
                              (esp_enc_alg == "cast") &&
                              (esp_auth_alg == "hmac-sha")
                           )
                           ||
                           ( (ah_present == "yes") &&
                             (
                               (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                               (local_filter_port == "80") ||
(remote_filter_port == "80")
                              ) &&
                              (ah_auth_alg == "hmac-md5")
                           )
                         )
                       )
                       ||
                       ( (security_level == "high") &&
                         (
                           ( (esp_present == "yes") &&
                             (
                               (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                               (local_filter_port == "6033") ||
(remote_filter_port == "6033")
                              ) &&
                              (esp_enc_alg == "3des") &&
                              (esp_auth_alg == "hmac-sha")
                           )
                           ||
                           ( (ah_present == "yes") &&
                             (
                               (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                               (local_filter_port == "80") ||
(remote_filter_port == "80")
                              ) &&
                              (ah_auth_alg == "hmac-sha")
                           )
                         )
                       )
                     )
                   )
                   ||
                   ( (network_mode == "impacted") &&
                     (
                       ( (security_level == "low") &&
                         (
                           ( (esp_present == "yes") &&
                             (
                               (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
```

```
                            (local_filter_port == "6033") ||
(remote_filter_port == "6033")
                           ) &&
                           (esp_enc_alg == "des") &&
                           (esp_auth_alg == "hmac-md5")
                         )
                         ||
                         ( (ah_present == "yes") &&
                           (
                             (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                             (local_filter_port == "80") ||
(remote_filter_port == "80")
                           ) &&
                           (ah_auth_alg == "hmac-md5")
                         )
                       )
                     )
                     ||
                     ( (security_level == "medium") &&
                       (
                         ( (esp_present == "yes") &&
                           (
                             (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                             (local_filter_port == "6033") ||
(remote_filter_port == "6033")
                           ) &&
                           (esp_enc_alg == "des") &&
                           (esp_auth_alg == "hmac-md5")
                         )
                         ||
                         ( (ah_present == "yes") &&
                           (
                             (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                             (local_filter_port == "80") ||
(remote_filter_port == "80")
                           ) &&
                           (ah_auth_alg == "hmac-md5")
                         )
                       )
                     )
                     ||
                     ( (security_level == "high") &&
                       (
                         ( (esp_present == "yes") &&
                           (
                             (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                             (local_filter_port == "6033") ||
(remote_filter_port == "6033")
                           ) &&
                           (esp_enc_alg == "3des") &&
                           (esp_auth_alg == "hmac-md5")
                         )
                         ||
```

```
                        ( (ah_present == "yes") &&
                          (
                            (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                            (local_filter_port == "80") ||
(remote_filter_port == "80")
                          ) &&
                          (ah_auth_alg == "hmac-sha")
                        )
                      )
                    )
                  )
                )
                ||
                ( (network_mode == "crisis") &&
                  (
                    ( (security_level == "low") &&
                      (
                        ( (esp_present == "yes") &&
                          (
                            (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                            (local_filter_port == "6033") ||
(remote_filter_port == "6033")
                          ) &&
                          (esp_enc_alg == "3des") &&
                          (esp_auth_alg == "hmac-sha")
                        )
                        ||
                        ( (ah_present == "yes") &&
                          (
                            (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                            (local_filter_port == "80") ||
(remote_filter_port == "80")
                          ) &&
                          (ah_auth_alg == "hmac-sha")
                        )
                      )
                    )
                    ||
                    ( (security_level == "medium") &&
                      (
                        ( (esp_present == "yes") &&
                          (
                            (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                            (local_filter_port == "6033") ||
(remote_filter_port == "6033")
                          ) &&
                          (esp_enc_alg == "3des") &&
                          (esp_auth_alg == "hmac-sha")
                        )
                        ||
                        ( (ah_present == "yes") &&
                          (
```

```
                          (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                          (local_filter_port == "80") ||
(remote_filter_port == "80")
                        ) &&
                        (ah_auth_alg == "hmac-sha")
                      )
                    )
                  )
                  ||
                  ( (security_level == "high") &&
                    (
                      ( (esp_present == "yes") &&
                        (
                          (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                          (local_filter_port == "6033") ||
(remote_filter_port == "6033")
                        ) &&
                        (esp_enc_alg == "aes") &&
                        (esp_auth_alg == "hmac-sha")
                      )
                      ||
                      ( (ah_present == "yes") &&
                        (
                          (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                          (local_filter_port == "80") ||
(remote_filter_port == "80")
                        ) &&
                        (ah_auth_alg == "hmac-sha")
                      )
                    )
                  )
                )
              )
              ||
              ( (network_mode == "default") &&
                (security_level == "default") &&
                (
                  ( (esp_present == "yes") &&
                    (
                      (local_filter_port == "23")   ||
(remote_filter_port == "23")   ||
                      (local_filter_port == "6033") ||
(remote_filter_port == "6033")
                    ) &&
                    (esp_enc_alg == "des") &&
                    (esp_auth_alg == "hmac-md5")
                  )
                  ||
                  ( (ah_present == "yes") &&
                    (
                      (local_filter_port == "79") ||
(remote_filter_port == "79") ||
                      (local_filter_port == "80") ||
(remote_filter_port == "80")
```

```
                    ) &&
                    (ah_auth_alg == "hmac-md5")
                  )
                )
              )
            )
          )
       -> "true";
```

### 2. Client-Side DSS Gateway

#### *a. /root/initialize_flows*

This script creates policy “flows” or rules for use by the ISAKMP subsystem under OpenBSD. These policy flows establish correct routing paths or tunnels for IPsec protection of outbound communication, and requirements for specific inbound communications to arrive with IPsec protection applied.

For additional protocols to be protected under the current DSS IPsec implementation, additional inbound and outbound flows must be defined in this file for each well-known port to be protected.

This script is executed by the client-side DSS start up script /root/isakmp.

---

```
#!/bin/sh
#############################################################################
# This script creates the flows that ISAKMPD will use with any security
# associations (SA) that it creates.  This script *must* be run before
this
# host will be able to *initiate* any IPsec-protected communications.
#############################################################################
#jfh: The only flows necessary are "net-to-net" (or gateway-to-gateway)
#     flows.  This is because we route *all* outbound packets to
#     192.168.0.0/24 (the MLS server addresses) through the loopback
before
#     putting them onto the wire with the command:
#
#     THIS NEEDS FIXING...
#     The TPE services on the MLS server's udp 6002 and 6102 might be
#     better-defined by specifying the TPE source port 6033.  This
might
#     get things screwed up since the client might occasionally use
port
#     6033 for something like http.  If a decision is made on this
issue,
#     this note can be removed.
#############################################################################
```

```
##############################################################################
# Path to binary
IPSECADM=/sbin/ipsecadm


##############################################################################
# Local and remote hosts, nets, netmasks
LOCAL_GATEWAY=192.168.1.100
REMOTE_GATEWAY=192.168.1.1

LOCAL_NET=192.168.2.0
REMOTE_NET=192.168.0.0

NETMASK_24=255.255.255.0
NETMASK_32=255.255.255.255


##############################################################################
# remove (flush) any current flows
#
$IPSECADM flush


##############################################################################
# Set-up flows for the two specific hosts
# Use for defining applications FINGER and TELNET
#  ESP for TELNET (tcp 23) and TPE (udp 6002, 6102)
#  AH for FINGER (tcp 79) and HTTP (tcp 80)
#    -dport for egress traffic
#    -sport for ingress traffic
#

#egress flow for finger
$IPSECADM flow -dst $REMOTE_GATEWAY -proto ah \
               -addr $LOCAL_NET $NETMASK_24 $REMOTE_NET $NETMASK_24 \
               -transport tcp -dport 79 \
             -src $LOCAL_GATEWAY -out -require

#ingress flow for finger
$IPSECADM flow -dst $REMOTE_GATEWAY -proto ah \
               -addr $REMOTE_NET $NETMASK_24 $LOCAL_NET $NETMASK_24 \
               -transport tcp -sport 79 \
               -src $REMOTE_GATEWAY -in -require

##############################################################################

#egress flow for telnet
$IPSECADM flow -dst $REMOTE_GATEWAY -proto esp \
               -addr $LOCAL_NET $NETMASK_24 $REMOTE_NET $NETMASK_24 \
               -transport tcp -dport 23 \
             -src $LOCAL_GATEWAY -out -require

#ingress flow for telnet
$IPSECADM flow -dst $REMOTE_GATEWAY -proto esp \
               -addr $REMOTE_NET $NETMASK_24 $LOCAL_NET $NETMASK_24 \
               -transport tcp -sport 23 \
               -src $REMOTE_GATEWAY -in -require
```

```
#########################################################################

#egress flow for http
$IPSECADM flow -dst $REMOTE_GATEWAY -proto ah \
               -addr $LOCAL_NET $NETMASK_24 $REMOTE_NET $NETMASK_24 \
               -transport tcp -dport 80 \
             -src $LOCAL_GATEWAY -out -require

#ingress flow for http
$IPSECADM flow -dst $REMOTE_GATEWAY -proto ah \
               -addr $REMOTE_NET $NETMASK_24 $LOCAL_NET $NETMASK_24 \
               -transport tcp -sport 80 \
               -src $REMOTE_GATEWAY -in -require

#########################################################################

#egress flow for TPE services, CLIENT PORT 6033
$IPSECADM flow -dst $REMOTE_GATEWAY -proto esp \
               -addr $LOCAL_NET $NETMASK_24 $REMOTE_NET $NETMASK_24 \
               -transport udp -sport 6033 \
             -src $LOCAL_GATEWAY -out -require

#ingress flow for TPE services, CLIENT PORT 6033
$IPSECADM flow -dst $REMOTE_GATEWAY -proto esp \
               -addr $REMOTE_NET $NETMASK_24 $LOCAL_NET $NETMASK_24 \
               -transport udp -dport 6033 \
               -src $REMOTE_GATEWAY -in -require

#########################################################################
exit 0
#########################################################################
#########################################################################
```

***b. /etc/isakmpd/isakmpd.conf***

This file contains the configuration of the client-side ISAKMP daemon used in IPsec. Specifications in this file include what IP address the daemon is to listen on for requests, what protocol is to be used to protect the key exchange negotiation process, and a listing of the server side DSS gateway “peer” with which the client-side daemon must communicate. The only lines likely to require changes for use by a second client and TPE are under the “General”, “Phase 1”, and “Peer” sections.

---

```
#########################################################################
# isakmpd.conf - configured for the TPE/client-side
#########################################################################
# CONOPS: The server side of the IPsec-protected tunnel sets policy
while the
#         client-side (this side) creates "flows" and attempts to
negotiate
```

```
#         a suite that is acceptable to the server's policy.
#
# The important thing in this file that (may) change is the list of
suites
# to try under "Default-Phase-2-Suites" and the IP-addresses of the
# endpoints.
###############################################################################

[General]
Listen-on=        192.168.1.100
Shared-SADB=            Defined
Retransmits=            5
Exchange-max-time=      120
Default-Phase-2-Suites= QM-ESP-DES-MD5-SUITE,QM-AH-MD5-SUITE,QM-ESP-
3DES-SHA-SUITE,QM-AH-SHA-SUITE,QM-ESP-CAST-SHA-SUITE,QM-ESP-3DES-MD5-
SUITE,QM-ESP-AES-SHA-SUITE

[Phase 1]
192.168.1.1=            Peer-192.168.1.1/192.168.1.100

[Peer-192.168.1.1/192.168.1.100]
Phase=                  1
Transport=        udp
Local-address=          192.168.1.100
Address=          192.168.1.1
Configuration=          Default-main-mode
Authentication=         mekmitasdigoat

[Default-main-mode]
DOI=              IPSEC
EXCHANGE_TYPE=          ID_PROT
Transforms=       3DES-SHA


###############################################################################
# No changes need to be made from here-forward
###############################################################################
# DES

[QM-ESP-DES-SUITE]
Protocols=        QM-ESP-DES

[QM-ESP-DES-PFS-SUITE]
Protocols=        QM-ESP-DES-PFS

[QM-ESP-DES-MD5-SUITE]
Protocols=        QM-ESP-DES-MD5

[QM-ESP-DES-MD5-PFS-SUITE]
Protocols=        QM-ESP-DES-MD5-PFS

[QM-ESP-DES-SHA-SUITE]
Protocols=        QM-ESP-DES-SHA

[QM-ESP-DES-SHA-PFS-SUITE]
Protocols=        QM-ESP-DES-SHA-PFS
```

```
# 3DES

[QM-ESP-3DES-SHA-SUITE]
Protocols=        QM-ESP-3DES-SHA

[QM-ESP-3DES-SHA-PFS-SUITE]
Protocols=        QM-ESP-3DES-SHA-PFS

# CAST (new Evie section)

[QM-ESP-CAST-SHA-SUITE]
Protocols=        QM-ESP-CAST-SHA

[QM-ESP-CAST-MD5-SUITE]
Protocols=        QM-ESP-CAST-MD5

[QM-ESP-CAST-SHA-PFS-SUITE]
Protocols=        QM-ESP-CAST-SHA-PFS

[QM-ESP-CAST-MD5-PFS-SUITE]
Protocols=        QM-ESP-CAST-MD5-PFS

# AH

[QM-AH-MD5-SUITE]
Protocols=        QM-AH-MD5

[QM-AH-MD5-PFS-SUITE]
Protocols=        QM-AH-MD5-PFS

[QM-AH-SHA-SUITE]
Protocols=        QM-AH-SHA

[QM-AH-SHA-PFS-SUITE]
Protocols=        QM-AH-SHA-PFS

# AH + ESP

[QM-AH-MD5-ESP-DES-SUITE]
Protocols=        QM-AH-MD5,QM-ESP-DES

[QM-AH-MD5-ESP-DES-MD5-SUITE]
Protocols=        QM-AH-MD5,QM-ESP-DES-MD5

[QM-ESP-DES-MD5-AH-MD5-SUITE]
Protocols=        QM-ESP-DES-MD5,QM-AH-MD5

# Quick mode protocols

# DES

[QM-ESP-DES]
PROTOCOL_ID=            IPSEC_ESP
Transforms=       QM-ESP-DES-XF

[QM-ESP-DES-MD5]
PROTOCOL_ID=            IPSEC_ESP
```

```
Transforms=       QM-ESP-DES-MD5-XF

[QM-ESP-DES-MD5-PFS]
PROTOCOL_ID=            IPSEC_ESP
Transforms=       QM-ESP-DES-MD5-PFS-XF

[QM-ESP-DES-SHA]
PROTOCOL_ID=            IPSEC_ESP
Transforms=       QM-ESP-DES-SHA-XF

# 3DES

[QM-ESP-3DES-SHA]
PROTOCOL_ID=            IPSEC_ESP
Transforms=       QM-ESP-3DES-SHA-XF

[QM-ESP-3DES-SHA-PFS]
PROTOCOL_ID=            IPSEC_ESP
Transforms=       QM-ESP-3DES-SHA-PFS-XF

[QM-ESP-3DES-SHA-TRP]
PROTOCOL_ID=            IPSEC_ESP
Transforms=       QM-ESP-3DES-SHA-TRP-XF

# CAST (new Evie section)

[QM-ESP-CAST-SHA]
PROTOCOL_ID=            IPSEC_ESP
Transforms=       QM-ESP-CAST-SHA-XF

[QM-ESP-CAST-MD5]
PROTOCOL_ID=            IPSEC_ESP
Transforms=       QM-ESP-CAST-MD5-XF

[QM-ESP-CAST-SHA-PFS]
PROTOCOL_ID=            IPSEC_ESP
Transforms=       QM-ESP-CAST-SHA-PFS-XF

[QM-ESP-CAST-MD5-PFS]
PROTOCOL_ID=            IPSEC_ESP
Transforms=       QM-ESP-CAST-MD5-PFS-XF

# AH MD5

[QM-AH-MD5]
PROTOCOL_ID=            IPSEC_AH
Transforms=       QM-AH-MD5-XF

[QM-AH-MD5-PFS]
PROTOCOL_ID=            IPSEC_AH
Transforms=       QM-AH-MD5-PFS-XF

# AH SHA

[QM-AH-SHA]
PROTOCOL_ID=            IPSEC_AH
Transforms=       QM-AH-SHA-XF
```

```
[QM-AH-SHA-PFS]
PROTOCOL_ID=            IPSEC_AH
Transforms=       QM-AH-SHA-PFS-XF

# Quick mode transforms

# ESP DES+MD5

[QM-ESP-DES-XF]
TRANSFORM_ID=           DES
ENCAPSULATION_MODE=     TUNNEL
Life=             LIFE_600_SECS

[QM-ESP-DES-MD5-XF]
TRANSFORM_ID=           DES
ENCAPSULATION_MODE=     TUNNEL
AUTHENTICATION_ALGORITHM=     HMAC_MD5
Life=             LIFE_600_SECS

[QM-ESP-DES-MD5-PFS-XF]
TRANSFORM_ID=           DES
ENCAPSULATION_MODE=     TUNNEL
GROUP_DESCRIPTION=      MODP_1024
AUTHENTICATION_ALGORITHM=     HMAC_MD5
Life=             LIFE_3600_SECS

[QM-ESP-DES-SHA-XF]
TRANSFORM_ID=           DES
ENCAPSULATION_MODE=     TUNNEL
AUTHENTICATION_ALGORITHM=     HMAC_SHA
Life=             LIFE_600_SECS

# 3DES

[QM-ESP-3DES-SHA-XF]
TRANSFORM_ID=           3DES
ENCAPSULATION_MODE=     TUNNEL
AUTHENTICATION_ALGORITHM=     HMAC_SHA
Life=             LIFE_60_SECS

[QM-ESP-3DES-SHA-PFS-XF]
TRANSFORM_ID=           3DES
ENCAPSULATION_MODE=     TUNNEL
AUTHENTICATION_ALGORITHM=     HMAC_SHA
GROUP_DESCRIPTION=      MODP_1024
Life=             LIFE_3600_SECS

[QM-ESP-3DES-SHA-TRP-XF]
TRANSFORM_ID=           3DES
ENCAPSULATION_MODE=     TRANSPORT
AUTHENTICATION_ALGORITHM=     HMAC_SHA
Life=             LIFE_60_SECS

#CAST (new Evie section)

[QM-ESP-CAST-SHA-XF]
```

```
TRANFORM_ID=                    CAST
ENCAPSULATION_MODE=             TUNNEL
AUTHENTICATION_ALGORITHM=     HMAC_SHA
Life=                     LIFE_60_SECS

[QM-ESP-CAST-MD5-XF]
TRANFORM_ID=                    CAST
ENCAPSULATION_MODE=             TUNNEL
AUTHENTICATION_ALGORITHM=     HMAC_MD5
Life=                     LIFE_60_SECS

[QM-ESP-CAST-SHA-PFS-XF]
TRANFORM_ID=                    CAST
ENCAPSULATION_MODE=             TUNNEL
AUTHENTICATION_ALGORITHM=     HMAC_SHA
GROUP_DESCRIPTION=              MODP_1024
Life=                     LIFE_60_SECS

[QM-ESP-CAST-MD5-PFS-XF]
TRANFORM_ID=                    CAST
ENCAPSULATION_MODE=             TUNNEL
AUTHENTICATION_ALGORITHM=     HMAC_MD5
GROUP_DESCRIPTION=              MODP_768
Life=                     LIFE_60_SECS

# AH

[QM-AH-MD5-XF]
TRANSFORM_ID=           MD5
ENCAPSULATION_MODE=     TUNNEL
AUTHENTICATION_ALGORITHM=     HMAC_MD5
Life=             LIFE_60_SECS

[QM-AH-MD5-PFS-XF]
TRANSFORM_ID=           MD5
ENCAPSULATION_MODE=     TUNNEL
GROUP_DESCRIPTION=      MODP_768
AUTHENTICATION_ALGORITHM=     HMAC_MD5
Life=             LIFE_3600_SECS

[QM-AH-SHA-XF]
TRANSFORM_ID=           SHA
ENCAPSULATION_MODE=     TUNNEL
AUTHENTICATION_ALGORITHM=     HMAC_SHA
Life=             LIFE_60_SECS

[QM-AH-SHA-PFS-XF]
TRANSFORM_ID=           SHA
ENCAPSULATION_MODE=     TUNNEL
GROUP_DESCRIPTION=      MODP_1024
AUTHENTICATION_ALGORITHM=     HMAC_SHA
Life=             LIFE_3600_SECS

[LIFE_30_SECS]
LIFE_TYPE=        SECONDS
LIFE_DURATION=          30,25:35
```

```
[LIFE_60_SECS]
LIFE_TYPE=        SECONDS
LIFE_DURATION=          60,45:120

[LIFE_180_SECS]
LIFE_TYPE=        SECONDS
LIFE_DURATION=          180,120:240

[LIFE_600_SECS]
LIFE_TYPE=        SECONDS
LIFE_DURATION=          600,450:720

[LIFE_3600_SECS]
LIFE_TYPE=        SECONDS
LIFE_DURATION=          3600,1800:7200

[LIFE_1000_KB]
LIFE_TYPE=        KILOBYTES
LIFE_DURATION=          1000,768:1536

[LIFE_32_MB]
LIFE_TYPE=        KILOBYTES
LIFE_DURATION=          32768,16384:65536

[LIFE_4.5_GB]
LIFE_TYPE=        KILOBYTES
LIFE_DURATION=          4608000,4096000:8192000
```

### *c. /etc/isakmpd/isakmpd.policy*

This file defines the KeyNote policy for protection of certain network communications based on the operational mode and security level of the network. Since the more restrictive policy definition is formulated on the server-side gateway, this policy file must be written in a manner that only specifies which protocols must receive AH protection and which must receive “ESP” protection.

---

```
KeyNote-Version: 2
Comment: This policy accepts ESP SAs from a remote that uses the right
password
Authorizer: "POLICY"
Licensees: "passphrase:mekmitasdigoat"
Conditions: app_domain == "IPsec policy" &&
          ( (esp_present == "yes") &&
              ( (
                   (local_filter_port == "23")   || (remote_filter_port
== "23")   ||
                   (local_filter_port == "6033") || (remote_filter_port
== "6033")
                )  &&
```

```
                ( (esp_enc_alg == "des") || (esp_enc_alg == "3des") ||
(esp_enc_alg == "aes") ||
                  (esp_enc_alg == "cast") || (esp_enc_alg ==
"blowfish") ) )
              ) ||
              ( (ah_present == "yes") &&
              ( (
                    (local_filter_port == "79") || (remote_filter_port ==
"79") ||
                    (local_filter_port == "80") || (remote_filter_port ==
"80")
                )  &&
                ( (ah_auth_alg == "hmac-md5") || (ah_auth_alg == "hmac-
sha") ||
                    (ah_auth_alg == "hmac-ripemd")   ) )
              )  -> "true";
```

### 3. TPE

#### *a. /root/net_config*

This script reconfigures the network interfaces (e.g. “eth0” and “eth1”) on the handheld TPE prototype to function in the Stage 4 environment. In this configuration, “eth0” is the MLS server side network interface, and “eth1” is the client-side network interface. After configuring the networking for the TPE, this script calls the “masq” script to configure the NAT functionality of the TPE.

---

```
#!/bin/sh

# source the environment
. /etc/profile

case $1 in
'start')
      echo "Stopping eth0"
      ifconfig eth0 down

      echo "Stopping eth1"
      ifconfig eth1 down

      echo "Starting eth0, setting to 192.168.2.11/24"
      ifconfig eth0 192.168.2.11 netmask 255.255.255.0 up

      echo "Starting eth1, setting to 192.168.3.1/24"
      ifconfig eth1 192.168.3.1 netmask 255.255.255.0 up

      #echo "Removing default route:192.168.2.1"
      #route delete default gw 192.168.1.1

      echo "Setting default route to client-side security
GW:192.168.2.1"
```

```
      route add default gw 192.168.2.1

      echo "About to create NAT rules"
      /root/masq start

      echo "NAT tables:"
      iptables -nL
      iptables -t nat -nL
      ;;
'stop')
        ;;
*)
        echo "usage: $0 { start | stop }"
        ;;
esac
```

***b. /root/masq***

This script reconfigures NAT operation on the handheld TPE prototype to function in the Stage 4 environment. "OIP" should be set to the IP address of the interface on the MLS server side network interface on TPE, and "IIP" should be set to the IP address of the client-side network interface.

---

```
#!/bin/sh

# source the environment
. /etc/profile

# OIP == outside IP address
OIP=192.168.2.11

# IIP == outside IP address
IIP=192.168.3.1

case $1 in
'start')
      echo "Configuring IP Masquerading..."

      # enable IP routing/forwarding
      echo 1 > /proc/sys/net/ipv4/ip_forward

      # flush existing rules
      iptables -F; iptables -t nat -F; iptables -t mangle -F

      # configure NAT
      iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to $OIP
      iptables -t nat -A PREROUTING -i eth0 -p udp --dport 6033 -j
ACCEPT
      iptables -t nat -A PREROUTING -i eth0 -p icmp --icmp-type ! 1 -j
ACCEPT
      iptables -t nat -A PREROUTING -i eth0 -j DNAT --to $IIP
```

```
      ;;
'stop')
        ;;
*)
        echo "usage: $0 { start | stop }"
        ;;
esac
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX D: TEST PROCEDURES

This appendix contains procedures for installation, configuration and testing of the Stages 3 and 4 implementations. Descriptions of the test plan coverage and the test plan report are included in Chapter IV, sections A and B, respectively.

## A. TEST FLOWCHART

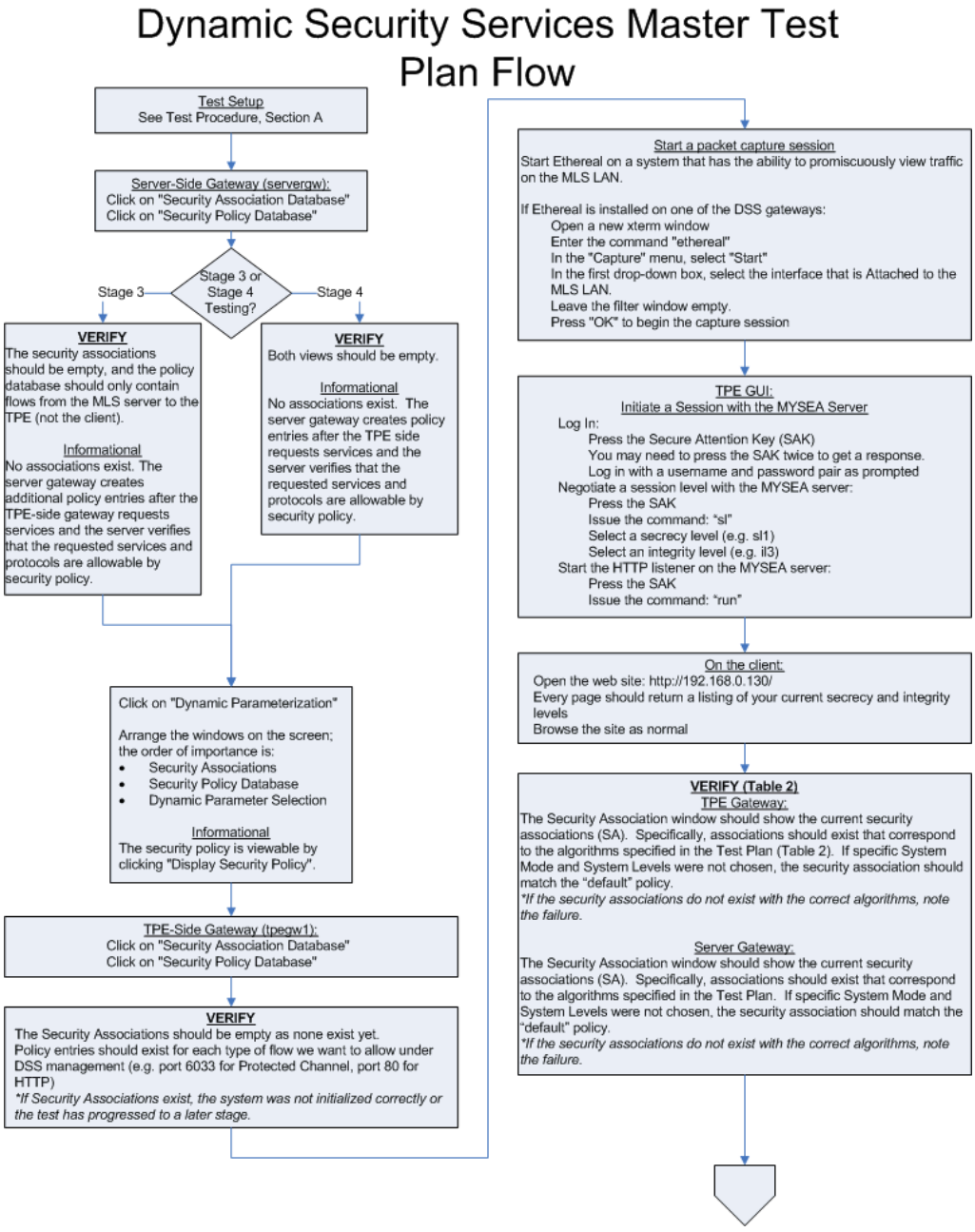


Figure 11. Test Procedure Flowchart, Part 1

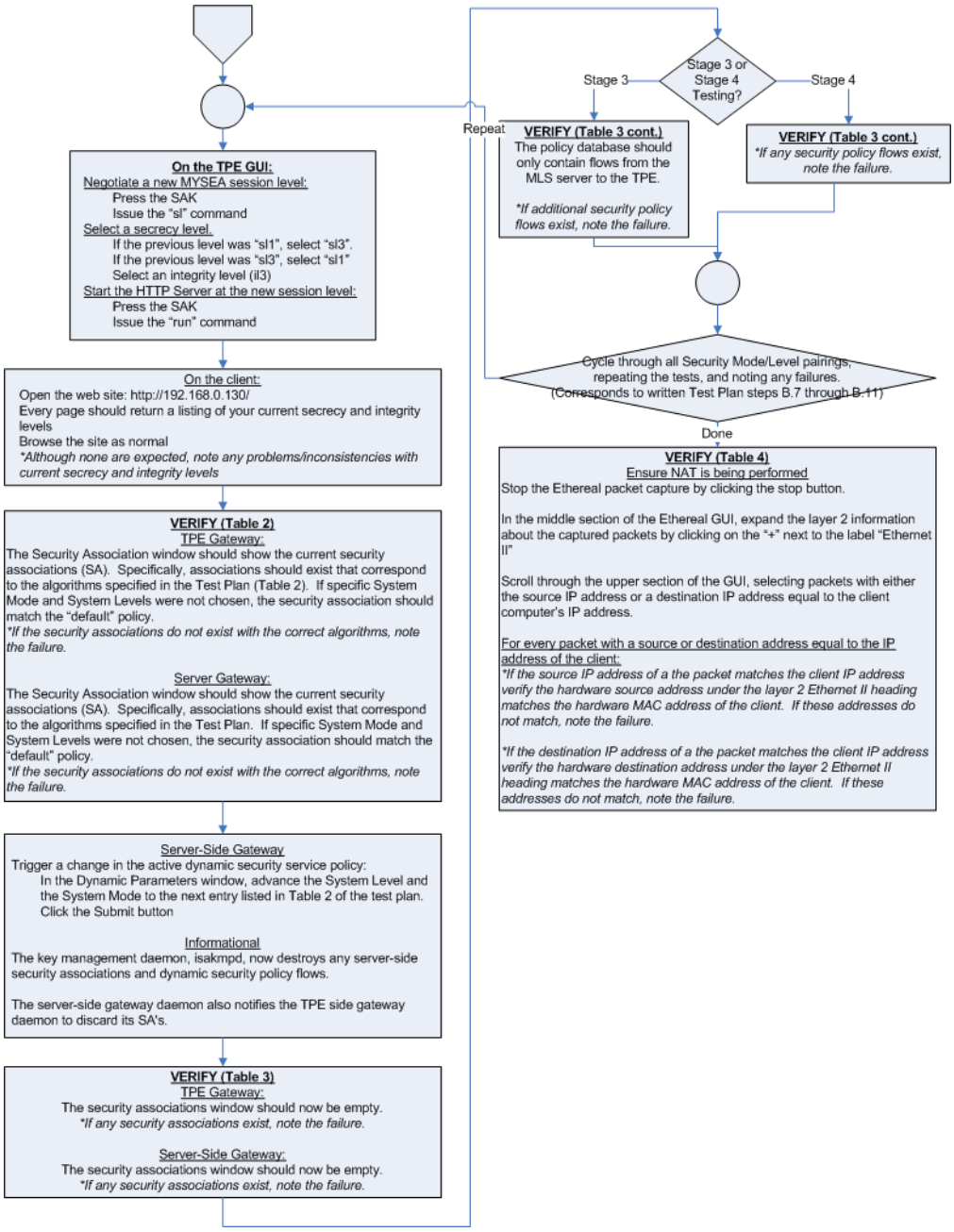


Figure 12. Test Procedure Flowchart, Part 2

## B. TEST PROCEDURE

A. DSS Test Setup

A.1. Setup the MYSEA server

A.1.1. Boot the MYSEA server

A.1.2. Log into the server

A.1.3. Switch to SL max:max

A.1.4. Issue the command "`startup`" to start the TPS daemon

A.2. Setup the OpenBSD Server-Side Gateway System (servergw)

A.2.1. Boot the system and login as root

A.2.2. Open an xterm window

A.2.3. Start the demonstration GUI by executing ./isakmp

A.2.4. Start the ISAKMP daemon by clicking "Start isakmpd"

A.2.5. Click on "Load Default DP" to load the default policy

A.2.6. Throughout the demo, isakmpd will generate syslog messages referencing "duplicate tags" and "negotiated SA failed policy check". These are remnant debugging statements inserted into isakmpd by the MYSEA development team and are not errors.

A.3. Setup the OpenBSD TPE-Side Gateway System (tpegw1)

A.3.1. Boot the system and login as root

A.3.2. Open an xterm window

A.3.3. Start the demonstration GUI by executing ./isakmp

A.4. Setup the TPE

A.4.1. If testing a stage 3 install:

A.4.1.1. Start the Java TPE by executing ./tpe in an xterm on the TPE-side gateway.

A.4.1.2. Click the portion of the TPE containing the IP address

A.4.1.3. If necessary, change the IP address to 192.168.0.130 and press ENTER

A.4.2. If testing a stage 4 install:

A.4.2.1. Cable the iPAQ TPE per the demonstration instructions and diagram in Appendix B. You must use the top Ethernet card for the connection to the TPE-Side Gateway and the lower card for the connection to the client

A.4.2.2. Reboot the iPAQ once the cabling is complete

A.4.2.3. Run the script "`/root/net_config start`" to initialize networking

A.4.2.4. Start the TPE GUI

A.4.3. The client machine can be any operating system.

A.4.3.1. If testing a stage 3 install, the client must:

A.4.3.1.1. Be configured with IP address 192.168.2.11

A.4.3.1.2. Have its default route set to 192.168.2.1 (TPE)

A.4.3.1.3. Have a web browser installed

A.4.3.2. If testing a stage 4 install, the client must:

A.4.3.2.1. Be configured with IP address 192.168.3.11

A.4.3.2.2. Have its default route set to 192.168.3.1 (TPE)

A.4.3.2.3. Have a web browser installed

B. Test Procedure

B.1. On the Server-Side Gateway (servergw)

B.1.1. Click on “Security Association Database”

B.1.2. Click on “Security Policy Database”

B.1.3. If testing a stage 3 install:

B.1.3.1. The security associations should be empty and the policy database should only contain flows from the MLS server to the TPE.

B.1.3.2. No associations exist, and the server gateway creates additional policy entries after the TPE side requests services and the server verifies that the requested services and protocols are allowable by security policy.

B.1.4. If testing a stage 4 install:

B.1.4.1. Both of these views should be empty.

B.1.4.2. No associations exist, and the server gateway creates policy entries after the TPE side requests services and the server verifies that the requested services and protocols are allowable by security policy.

B.1.5. The security policy is viewable by clicking “Display Security Policy”.

B.1.6. Click on “Dynamic Parameterization”

B.1.7. Arrange the windows on the screen; the order of importance is:

Security Associations

Security Policy Database

Dynamic Parameter Selection

B.2. On the TPE-Side Gateway (tpegw1)

B.2.1. Click on “Security Association Database”

B.2.2. Click on “Security Policy Database”

B.2.3. The Security Associations should be empty as none exist yet.

B.2.4. The policy entries should exist for each type of flow we want to allow under DSS management.

B.3. Start a packet capture session for this test.

B.3.1. Start Ethereal on a system that has the ability to promiscuously view traffic on the MLS LAN.

B.3.2. If Ethereal is installed on one of the DSS gateways:

B.3.2.1. Open a new xterm window

B.3.2.2. Enter the command "`ethereal`"

B.3.2.3. In the "Capture" menu, select "Start"

B.3.2.4. In the first drop-down box, select the interface that is attached to the MLS LAN.

B.3.2.5. Leave the filter window empty.

B.3.2.6. Press "OK" to begin the capture session.

B.4. Start a Session with the MYSEA Server

B.4.1. In the TPE GUI:

B.4.1.1. Log In:

B.4.1.1.1. Press the Secure Attention Key (SAK)

B.4.1.1.2. You may need to press the SAK twice to get a response.

B.4.1.1.3. Log in with a username and password pair as prompted

B.4.1.2. Negotiate a session level with the MYSEA server:

B.4.1.2.1. Press the SAK

B.4.1.2.2. Issue the "sl" command

B.4.1.2.3. Select a secrecy level (e.g. sl1)

B.4.1.2.4. Select an integrity level (e.g. il3)

B.4.1.3. Start the application (Apache) listener on the MYSEA server:

B.4.1.3.1. Press the SAK

B.4.1.3.2. Issue the "run" command

B.5. On the client:

B.5.1. Open the web site: http://192.168.0.130/

Every page should return a listing of your current secrecy and integrity levels

B.5.2. Browse the site as normal

B.6. VERIFY:

B.6.1. On the TPE gateway:

B.6.1.1. The Security Association window should show the current security associations (SA). Specifically, associations should exist that correspond to the algorithms specified in the Test Plan. If specific

System Mode and System Levels were not chosen, the security association should match the “default” policy.

B.6.1.2. *If the security associations do not exist with the correct algorithms, note the failure.

B.6.2. On the server gateway:

B.6.2.1. The Security Association window should show the current security associations (SA). Specifically, associations should exist that correspond to the algorithms specified in the Test Plan. If specific System Mode and System Levels were not chosen, the security association should match the “default” policy.

B.6.2.2. *If the security associations do not exist with the correct algorithms, note the failure.

B.7. On the TPE, negotiate a new MYSEA session level:

B.7.1. Press the SAK

B.7.2. Issue the “sl” command

B.7.3. Select a secrecy level.

B.7.3.1. If the previous level was “sl1”, select “sl3”.

B.7.3.2. If the previous level was “sl3”, select “sl1”

B.7.4. Select an integrity level (il3)

B.8. Restart the application (Apache) listener on the MYSEA server:

B.8.1. Press the SAK

B.8.2. Issue the “run” command

B.9. On the client:

B.9.1. Open the web site: http://192.168.0.130/

Every page should return a listing of your current secrecy and integrity levels. Although none are expected, note any problems/inconsistencies with current secrecy and integrity levels.

B.9.2. Browse the site as normal.

B.10. On the server-side gateway, trigger a change in the security service policy:

B.10.1. In the Dynamic Parameters window, advance the System Level and the System Mode to the next entry listed in the Test Plan.

B.10.2. Click the Submit button

B.10.2.1. The key management daemon, isakmpd, now destroys any server-side security associations and dynamic security policy flows.

B.10.2.2. The server-side gateway daemon also notifies the TPE side gateway daemon to discard its SA’s.

B.11. VERIFY:

B.11.1. On the TPE gateway:

B.11.1.1. The security associations window should now be empty.

B.11.1.2. *If any security associations exist, note the failure.

B.11.2. On the server gateway:

B.11.2.1. The security associations window should now be empty.

B.11.2.2. *If any security associations exist, note the failure.

B.11.2.3. If testing a stage 3 install:

B.11.2.3.1. The policy database should only contain flows from the MLS server to the TPE.

B.11.2.3.2. *If additional security policy flows exist, note the failure.

B.11.2.4. If testing a stage 4 install:

B.11.2.4.1. *If any security policy flows exist, note the failure.

B.12. Repeat steps B.7 through B.11, cycling through all Security Mode/Level pairings, and noting any failures.

C. Review the packet capture

C.1. Stop the Ethereal packet capture by clicking the stop button.

C.2. VERIFY (Table 4):

C.2.1. Ensure NAT is being performed. (No traffic appears on the MLS LAN with a source or destination address set to the client's address.)

C.2.1.1. Stop the Ethereal packet capture by clicking the stop button.

C.2.1.2. In the middle section of the Ethereal GUI, expand the layer 2 information about the captured packets by clicking on the "+" next to the label "Ethernet II"

C.2.1.3. Scroll through the upper section of the GUI, selecting packets with either the source IP address or a destination IP address equal to the client computer's IP address.

C.2.1.4. For every packet with a source or destination address equal to the IP address of the client:

C.2.1.4.1. *If the source IP address of a the packet matches the client IP address verify the hardware source address under the layer 2

Ethernet II heading matches the hardware MAC address of the client. If these addresses do not match, note the failure.

C.2.1.4.2. *If the destination IP address of a the packet matches the client IP address verify the hardware destination address under the layer 2 Ethernet II heading matches the hardware MAC address of the client. If these addresses do not match, note the failure.

# LIST OF REFERENCES


[BEL76] Bell, D. E. & La Padula, L. J. (1976). Secure Computer System: Unified Exposition and Multics Interpretation. ESD-TR-75-306. Mitre Corporation, Bedford, MA.

[BIB77] Biba, K .J. (1977). Integrity Considerations for Secure Computer Systems. ESD-TR-76-372. Mitre Corporation, Bedford, MA.

[BLA99] Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A. (1999, September). Request for Comments: 2704 – The KeyNote Trust-Management System Version 2

[BLA01] Blaze, M., Ioannidis, J., Keromytis, A. (2001, February). "Trust Management for IPsec." Proceedings of the Internet Society Symposium on Network and Distributed Systems Security (SNDSS) 139 - 151.

[CC04] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, Version 2.2, Revision 256 – CCIMB-2004-01-001 (2004, January) Available: http://www.commoncriteriaportal.org/public/files/ccpart1v2.2.pdf. Accessed: 03/25/2005.

[CYG04] CygnaCom Solutions (2004). Available: http://www.cygnacom.com/labs/pfSEL0181xts400.htm. Accessed: 02/12/2005.

[HAR98] Harkins, D., Carrel, D. (1998, November). Request for Comments: 2409 – The Internet Key Exchange (IKE). Available: ftp://ftp.rfc-editor.org/in-notes/rfc2409.txt. Accessed:03/26/2005.

[IRV00] Irvine, C. E., Levin, T. E. (2000, September). Quality of Security Service. Proceedings of the New Security Paradigms Workshop, Ballycotton, Ireland, 18-22

[IRV04] Irvine, C. E., Levin, T. E., Nguyen, T. D., Shifflett, D., Khosalim, J., Clark, P. C., Wong, A., Afinidad, F., Bibighaus, D., & Sears, J.(2004). "Overview of a High Assurance Architecture for Distributed Multilevel Security." Proceedings of the 5th IEEE Systems, Man and Cybernetics Information Assurance Workshop,38-45.

[KEN98] Kent, S., Atkinson, R. (1998, November). Request for Comments: 2401 – Security Architecture for the Internet Protocol. Available: ftp://ftp.rfc-editor.org/in-notes/rfc2401.txt. Accessed 03/26/2005.

[LEO00] de Leon, R. (2000, August). DOD CIO G&PM "GIG Network Operations". Available: http://www.defenselink.mil/nii/org/cio/doc/gig10-8460-082400.pdf. Accessed 03/27/2005

[MAU98] Maughan, D., Schertler, M., Schneider, M., Turner, J. (1998, November). Request for Comments: 2408 – Internet Security Association and Key Management Protocol (ISAKMP). Available: ftp://ftp.rfc-editor.org/in-notes/rfc2408.txt. Accessed:03/31/2005.

[NIA04] National Information Assurance Partnership (NIAP) (2004, December). Available: http://niap.nist.gov/cc-scheme/st/ST_VID3012.html. Accessed: 02/12/2005.

[OPE04] OpenBSD (2005, March). "13 – Using IPsec (IP Security Protocol)", The OpenBSD FAQ. Available: http://www.openbsd.org/faq/faq13.html. Accessed: 02/10/2005.

[OPE05] OpenBSD (2005, March). Available: http://www.openbsd.org. Accessed: 03/31/2005

[ORM96] Orman, H., (1998, November). Request for Comments: 2412 – The Oakley Key Determination Protocol. Available: ftp://ftp.rfc-editor.org/in-notes/rfc2412.txt. Accessed: 03/31/2005.

[SEA04] Sears, J. D. (2004, September). Simultaneous Connection Management and Protection in a Distributed Multilevel Security Environment. Master's Thesis, Naval Postgraduate School, Monterey, CA.

[SYP02a] Sypropoulou, E., Agar, C., Levin, T., & Irvine, C. (2002, March) “IPsec Modulation for Quality of Security Service.” Proceedings of the International System Security Engineering Association Conference, Orlando Florida

[SYP02b] Sypropoulou, E., Levin, T., & Irvine, C. (2002, September) “Demonstration of Quality of Security Service Awareness for IPsec” Center for INFOSEC Studies and Research whitepaper # NPS-CS-02-005 Available: http://cisr.nps.navy.mil/downloads/QoSS_Ipsec_Demo_WP.pdf. Accessed: 03/31/2005

THIS PAGE INTENTIONALLY LEFT BLANK

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Hugo A. Badillo
   NSA
   Fort Meade, MD

4. George Bieber
   OSD
   Washington, DC

5. Deborah Cooper
   DC Associates, LLC
   Roslyn, VA

6. CDR Daniel L. Currie
   PMW 161
   San Diego, CA

7. CDR James Downey
   NAVSEA
   Washington, DC

8. Dr. Diana Gant
   National Science Foundation
   Arlington, VA

9. Lewis Gutman
   SPAWAR Systems Center
   San Diego, CA

10. Richard Hale
    DISA
    Falls Church, VA

11. LCDR Scott D. Heller
    SPAWAR
    San Diego, CA

12. John Horn
Naval Postgraduate School
Monterey, CA

13. Dr. Cynthia E. Irvine
Naval Postgraduate School
Monterey, CA

14. Russell Jones
N641
Arlington, VA

15. Wiley Jones
OSD
Washington, DC

16. David Ladd
Microsoft Corporation
Redmond, WA

17. Steve LaFountain
NSA
Fort Meade, MD

18. Dr. Carl Landwehr
National Science Foundation
Arlington, VA

19. Dr. Greg Larson
IDA
Alexandria, VA

20. Penny Lehtola
NSA
Fort Meade, MD

21. Ernest Lucier
Federal Aviation Administration
Washington, DC

22. Dr. Vic Maconachy
NSA
Fort Meade, MD

23. Doug Maughan
Department of Homeland Security
Washington, DC

24. John Mildner
SPAWAR
Charleston, SC

25. Dr. John Monastra
Aerospace Corporation
Chantilly, VA

26. Thuy Nguyen
Naval Postgraduate School
Monterey, CA

27. Steve Rose
BAE Systems
Herndon, VA

28. Keith Schwalm
Good Harbor Consulting, LLC
Washington, DC

29. RADM Andrew Singer
NETWARCOM
Fort Meade, MD

30. Dr. Ralph Wachter
ONR
Arlington, VA

31. David Wirth
N641
Arlington, VA

32. Daniel Wolf
NSA
Fort Meade, MD

33. William Wolfe
SPAWAR Systems Center
San Diego, CA

34. James Yerovi
National Reconnaissance Organization
Chantilly, VA

35. CAPT Robert Zellmann
CNO Staff N614
Arlington, VA